sitepoint

THINKING WEB
VOICES OF THE COMMUNITY

THE SITEPOINT COMMUNITY

TAP INTO THE VAST WEALTH OF KNOWLEDGE OF THE COMMUNITY

www.it-ebooks.info

Summary of Contents

Preface xvii
1. Anatomy of a Website 1
2. Designing in the Dark 19
3. Everything Must Go! 31
4. Going Freelance 43
5. Successful PSD to HTML Freelancing 49
6. Write Email Markup That Doesn't Explode in the Inbox 73
7. Make Your Website Stand Out from the Crowd 85
8. Information Organization and the Web 97
9. Using Vector Graphics to Build a Noughts & Crosses Game 105
10. Efficient ActionScript 157
11. Databases: The Basic Concepts 169
12. The Iceberg of TCP/IP 187

www.it-ebooks.info

THINKING WEB

VOICES OF THE COMMUNITY

BY **JOHN BORDA**
URSULA COMEAU
SHERRY CURRY
ALEX DAWSON
COYOTE HOLMBERG
RALPH MASON
PAUL O'BRIEN
CHRISTIAN SNODGRASS
ROBERT WELLOCK
CLIVE WICKHAM
NURIA ZUAZO

www.it-ebooks.info

Thinking Web: Voices of the Community

by John Borda, Ursula Comeau, Sherry Curry, Alex Dawson, Coyote Holmberg, Ralph Mason, Paul O'Brien, Christian Snodgrass, Robert Wellock, Clive Wickham, and Nuria Zuazo

Program Director: Lisa Lang
Technical Editor: Louis Simoneau
Editor: Kelly Steele
Cover Design: Alex Walker
Community Manager: Sarah Hawk

Printing History:

First Edition: March 2011

Notice of Rights

Some rights reserved. This book is provided under a Creative Commons Attribution-Noncommercial-ShareAlike 3.0 license [http://creativecommons.org/licenses/by-nc-sa/3.0/]. You are free to share or adapt this work on the condition that it is attributed to the SitePoint community, that your use is not commercial in nature, and that any derivative works are distributed under the same license.

Notice of Liability

The author and publisher have made every effort to ensure the accuracy of the information herein. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors and SitePoint Pty. Ltd., nor its dealers or distributors, will be held liable for any damages caused either directly or indirectly by the instructions contained in this book, or by the software or hardware products described herein.

Trademark Notice

Rather than indicating every occurrence of a trademarked name as such, this book uses the names only in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark.

Published by SitePoint Pty. Ltd.

48 Cambridge Street, Collingwood
VIC 3066 Australia
Web: www.sitepoint.com
Email: business@sitepoint.com

ISBN 978-0-9870908-9-8
Printed and bound in the United States of America

www.it-ebooks.info

About John Borda

John Borda has been designing websites for over 15 years, the last three as a freelancer at Bordaline Web Design (www.bordaline.co.uk). Originally from Gibraltar, he now lives in Newmarket, UK with his wife and three children. In 2009, John won the Newmarket Business Association's Community Engagement award for his involvement with a number of charities and nonprofit groups.

About Ursula Comeau

Ursula Tathiana Comeau, nicknamed the WP Gal, is a self-taught WordPress addict who loves everything to do with WordPress, blogging, and social media. She has a strong background in service, support, and technical training, and has been using computers and technology since the mid-80s. Born in Brazil, Ursula enjoyed a stint in the US from 1985 to 1988, before moving to Canada to live permanently. Ursula has been an active internet citizen since 1995.

You can drop by and say hi to her at http://ucwebcreations.com/, and if you'd like to hear her music, visit http://tathiana.com/.

About Sherry Curry

Sheryl Moore Curry is an assistant professor of library science and Head of Information & Web Technology Services at Edith Garland Dupré Library, University of Louisiana at Lafayette (USA). She has an MLIS degree from Louisiana State University, and is editor of the review column for the journal *Louisiana Libraries* and co-author of *Newbery & Caldecott Awards: A Subject Index* (Linwood Book Group, 2003). Sherry has written articles and book chapters on a variety of topics, as well as presented at national conferences on first-time publishing, web design, popular technological innovation choices, and the relationship between systems and services.

About Alex Dawson

Alexander Dawson (@AlexDawsonUK) is an award-winning, self-taught, freelance web professional, writer, published author, and recreational software developer from Brighton, England. With more than 10 years of industry experience, Alex spends his days running his consultancy firm HiTechy (http://www.hitechy.com/), writing professionally about web design, and giving his free time to assist others in the field.

www.it-ebooks.info

About Coyote Holmberg

Coyote Holmberg is a web producer who has been indulging her passion for web design and development since 1998. Currently, Coyote is the web production lead for the marketing team at American Greetings Interactive, where she implements and advises on markup, code, and content for the company's sites, email newsletters, and search optimization initiatives.

Coyote lives in Cleveland, Ohio with her husband Brian, three cats, two lizards, and a garden that threatens to take over her life.

About Ralph Mason

Ralph Mason spent 10 years as a classroom teacher before turning his hand to freelance web design. He posts tips on web design at his site pageaffairs.com, and is currently working on a series of free ebooks on various web topics for his site booksforlearning.com. In his spare time, he likes to swim, chop wood, edit classic children's stories, and contemplate life's deeper questions.

About Paul O'Brien

Paul O'Brien is a freelance web designer specializing in CSS layouts. He entered the world of web design back in 1998, and what started as a hobby soon became a full-time occupation. You'll often find Paul giving advice in the SitePoint forums, where he has racked up nearly 20,000 posts—all of them CSS-related.

About Christian Snodgrass

Christian Snodgrass is the owner of Azure Ronin Web Design (http://www.arwebdesign.net/) and an avid software developer. Currently, he is involved in developing an ActionScript 3 game engine and framework, which will be openly available in the near future. Christian also teaches and does curriculum development for iD Tech Camps (http://www.internaldrive.com), North America's #1 Summer Computer Camp.

About Robert Wellock

Robert J. Wellock hails from Yorkshire, England, where his family has over 400 years of experience in Agricultural Science. Robert describes himself as eccentric and has an appreciation for extremely dry humor; he metaphorically wears an casual, unassuming old robe. Unconventional in many respects, Robert has an affinity with XHTML syntax. He has a university background in Applied Biological Sciences, and more qualifications in computing and net-

www.it-ebooks.info

working than he can remember. In his spare time when not feeding the beasts, he plays with (X)HTML semantics, accessibility, and cascading style sheets, and lives at http://www.xhtmlcoder.com/.

About Clive Wickham

Clive Wickham is a software developer based near London, UK. Having completed a BSc in Software Engineering in 2004, he now works for a specialist engineering consultancy developing an electrical power system analysis software suite called ERACS. Clive's personal website can be found at: http://clive.wickhub.co.uk/.

About Nuria Zuazo

Nuria Zuazo is a software teacher and freelancer. She has been doing intensive training courses for the past five years, and has been interested in the Web since the very beginning. Nuria is a self-teacher, where she loves to try new things and figure out how they work, experimenting in the broadest sense of the word. She's a Jill of all trades, with curiosity driving her here and there.

www.it-ebooks.info

www.it-ebooks.info

Table of Contents

Preface xvii

Conventions Used in This Book xix

Code Samples xix

Tips, Notes, and Warnings xx

Chapter 1 Anatomy of a Website 1

Files and Folders 1

Hosting 1

Getting Your Site Online 2

The Root Folder 3

Optional Root Folder Assets 4

Beyond the Root Folder 6

Linking Everything Together 7

Anatomy of a Web Page 7

The `<head>` Section 10

The `<body>` Section 12

Beyond Static Sites 16

Enter the CMS 16

The Database 16

Conclusion 17

Chapter 2 Designing in the Dark 19

The Case of the Unknown Visitor 19

Setting the Scene 20

Who is the visitor? 20

You're the Detective 21

www.it-ebooks.info

A Challenge Ahead 22
Why It Really Matters 23
Clues, Evidence, and Theories 24
Great Expectations 24
Innovative Ideas 25
Building Visitor Solutions 26
Examine the Problems 26
Implement Experiences 27
Learning from Experience 28
Trial and Error 28
Convention Coding 29
Seeing the Light 30

Chapter 3 Everything Must Go! 31
The Hypertext Apocalypse 31
Crippling Decisions 32
Accessibility Matters 33
Senseless Dependence 34
Disabled Mediums 34
The Curse of Plugins 35
Accessible Assassinations 36
Spraying Weed Killer 36
Content Matters! 37
Progressive Enhancement 38
Starting from Scratch 38
Bulletproof Layering 39
It Matters to Your Users 40
The Risk of Dependence 40
The Future of the Fallen 41
Everything Must Go! 42

www.it-ebooks.info

Chapter 4 Going Freelance 43
Scenario One: Reception 43
Share a Different Perspective 44
Scenario Two: Shared Interest 44
Make Your Enemy Your Friend 45
A Second Pair of Eyes 46
Keep It Simple 46
Work for Nothing! 46

Chapter 5 Successful PSD to HTML Freelancing 49
Before You Start Coding 50
Rule 1: Always See the Work First Before Quoting 50
Rule 2: Establish What Kind of Work Is Required 51
Rule 3: Discover What Browser Support Is Needed 52
Rule 4: Verify the Timescales 54
Confirm How Much the Job Will Cost 54
Tools of the Trade 56
Getting Started 58
Getting Ready to Style 60
Structure Your Page Well 61
Slicing and Dicing 62
Where to start? 64
Graphic Design Considerations 65
Final Thoughts 71

Chapter 6 Write Email Markup That Doesn't Explode in the Inbox 73
HTML Email: the Power and the Glory 73

www.it-ebooks.info

Coding and Sending: Danger Awaits 74
Email Design and Content: The Basics 75
Subject Line 75
Call to Action 75
Content Positioning 75
Email Isn't the Web 76
Coding Like It's 1999 77
Step 1: Use Nested HTML Tables for Layout 77
Step 2: Writing Your CSS and Moving It Inline. 78
Step 3: Text Formatting and Paragraph Spacing 79
Adding Images and Animation 80
Avoid Using Background Images 80
Prepare for Image Blocking and Missing Images 81
Go Ahead, Include Your Dancing Hamster Animations 81
Beware Dynamic Content 82
It's Coded. Now What? 82
Resources 83

Chapter 7 **Make Your Website Stand Out from the Crowd** 85

So what exactly is social media? 86
Where does blogging fit into all of this? 87
Driving Traffic 88
Being Memorable 89
Rules of Engagement in Blogging 90
Is having a blog absolutely necessary? 91
The Role of SEO 92
Setting Up a Blog 94
In Summary 96

www.it-ebooks.info

Chapter 8 Information Organization and the Web 97
Getting Started 97
Word Choice 99
What does that term mean? 99
Selected Resources on Terminology 99
Navigation 100
Other Navigation Tools 101
Dead Ends 101
Before Launching 101
Usability 101
Review of Textual Content 102
Miscellaneous Information 103
The End is (Never) Near 103

Chapter 9 Using Vector Graphics to Build a Noughts & Crosses Game 105
What is Noughts and Crosses? 105
Vector Graphics 106
Vector Graphics with SVG 107
Vector Graphics with HTML5 Canvas 108
Building a Simple Game of Noughts & Crosses 110
Indispensable JavaScript 112
Creating an HTML Template 112
Embedding an External SVG Document 114
Inserting an HTML5 canvas Element 114
Creating the SVG Blueprint 114
Defining the Building Blocks for Our Game 115
Drawing the Playing Grid 117

www.it-ebooks.info

Defining the SVG Nought 119
Drawing a Nought 120
Defining the SVG Cross 123
Drawing a Cross 124
Defining the SVG Strikethrough 127
Drawing a Strikethrough 127
Defining the SVG "Game Over" Text 130
Drawing the Game Over Text 132
Developing the Game Logic 137
The `initialiseGame` Function 141
The `newGame` Function 144
The `canvasOnClick` Function 145
The `clearCanvas` Function 146
The `getCursorPosition` Function 148
The `isEmptyCell` Function 148
The `threeInALine` Function 148
The `endGame` Function 151
Summary 151
Taking It Further 152
Resources 154
SVG Tools 155

Chapter 10 Efficient ActionScript 157
Object-oriented ActionScript 157
Screens 158
Use a Single Enter Frame Event Listener 158
Singleton Pattern 159
Globalized Stage 162
Centralized Keyboard Input 164
Conclusion 166

www.it-ebooks.info

Chapter 11 Databases: The Basic Concepts 169
So, what exactly is a database? 169
Relational Databases 170
Some Terminology 171
Types of Relationships 173
More on Normalization 176
Database Design 179
Which database to choose? 180
MySQL and SQL Instructions 181
Some Good Practices 181
SQL Basic Syntax 182
Creating a Database 182
Creating a Table 183
Changing the Structure of a Table 183
Inserting Data into a Table 184
Modifying a Record 185
Reading Data 185

Chapter 12 The Iceberg of TCP/IP 187
What is TCP/IP? 187
Brief History of TCP/IP 188
The OSI Model Compared to the Internet Model 189
The TCP/IP Network Model 191
TCP: A Connection-orientated Protocol 192
IP: A Connectionless Protocol 193
Routers 193
IP Addressing and Subnets (Subnetting) 194
Addressing with IP Numbers 196
Class A, B, and C Addresses 198
Subnetting 198

www.it-ebooks.info

Port Numbers 200
Handshaking 201
The Different TCP/IP Protocols 201
User Datagram Protocol (UDP) 202
Domain Name Services (DNS) 202
Internet Control Message Protocol (ICMP) 202
File Transfer Protocol (FTP) 203
Hypertext Transfer Protocol (HTTP) 204
Dynamic Host Configuration Protocol (DHCP) 204
Simple Network Management Protocol (SNMP) 204
Telnet 205
Simple Mail Transfer Protocol (SMTP) 206
Post Office Protocol (POP)—Version 3 206
Internet Messaging Access Protocol (IMAP)—Version 4 207
Synchronous/Asynchronous and Duplex/Simplex Data Transmission 207
Summary 208

www.it-ebooks.info

Preface

In every SitePoint book's preface, we have a section called "The SitePoint Forums." It describes the forums as a place where you can ask questions on anything related to web design, development, hosting, and online marketing. It goes on to describe how the forums work: "some people ask, some people answer, and most people do a bit of both." I love that line. But what resonates the most for me is the part that says "sharing your knowledge benefits others and strengthens the community." Obviously, I have a greater investment in the strength of our community than most, but what I like about this description is that it makes it about everyone; combined strength.

And that's what this book is all about.

Over the years that I've been involved with the SitePoint Forums, I've heard people comment numerous times on the wealth of knowledge and talent that we have in the community. What better way to harness that knowledge than write a book?

As I sit here writing this introduction, I have no idea how the book will turn out. That's part of what makes the whole idea so exciting. Just like any online community, there are so many rogue factors that it's impossible to predict an outcome. What I do know is that regardless of whether you're a rookie to the world of web development, or you've been around the block a few times, there's bound to be something in here that's new to you. That's how communities work.

Many of the contributors to this book are considered experts in their fields, and I'm proud to have them as members of our community. What I'm even more pleased about is that they're forum members for no reason other than to pass that knowledge on. A lot of those experts weren't experts when they first came to us. In many cases—mine included—they showed up looking for answers to basic questions. If you stick around SitePoint long enough, you just might find yourself learning something new. It sneaks up on you. I'm unsure of when I realized that I actually knew stuff myself. One day, I noticed that my colleagues were coming to me with questions. It's a good feeling.

The twist to this book project is that a huge number of our forum members actually came to us by way of a SitePoint book. If you are one of those members, you'll un-

www.it-ebooks.info

derstand the value of having the backing of a community when you just can't make that damn code work. So now we're paying it forward.

If you've never ventured as far as the forums, consider this book your invitation. You're in for a treat. It might take a while for the value of that treat to become apparent—it can be a daunting place—but persevere, it'll be worth it.

The most important thing to keep in mind when visiting ANY forum for the first time is that the rest of the members are just people, and they all started somewhere. There are no stupid questions, and no mistakes that can't be rectified. I'd love to regale you with an entertaining anecdote of my first time visiting SitePoint, only I'd have to make one up. Like a great many of our members, I came by way of Google, and never left. I found code snippets, searched for answers to questions that I was too scared to ask, and laughed with some of the characters I met in the General Chat forum. I spent some time as a *lurker* (forum-speak for a person that hangs around and reads without actually posting) before my natural urge to show off got the better of me. I've never looked back.

I should issue you a warning, though. SitePoint can be addictive. It only took a couple of weeks, and I was hooked. I clocked up hours and hours on the forums, asking and answering questions, but also just playing the fool and making friends. It's a great place to let your hair down while bouncing new ideas off your peers. These days, I consider myself to be one of the luckiest people around, because I'm being paid to do something that I'd willingly do for free—in fact, I was!

So sit back and enjoy this book—the fruits of our labor. When you've finished reading, I looking forward to seeing you over at the forums. There's always room for one more.

—Sarah Hawk (aka HAWK)

www.it-ebooks.info

Conventions Used in This Book

You'll notice that we've used certain typographic and layout styles throughout the book to signify different types of information. Look out for the following items.

Code Samples

Code in this book will be displayed using a fixed-width font, like so:

```
<h1>A Perfect Summer's Day</h1>
<p>It was a lovely day for a walk in the park. The birds
were singing and the kids were all back at school.</p>
```

When an example requires multiple different files, the name of the file will appear at the top of the program listing, like this:

example.css

```
.footer {
  background-color: #CCC;
  border-top: 1px solid #333;
}
```

If only part of the file is displayed, this is indicated by the word *excerpt*:

example.css *(excerpt)*

```
  border-top: 1px solid #333;
```

If additional code is to be inserted into an existing example, the new code will be displayed in bold:

```
function animate() {
  new_variable = "Hello";
}
```

Where existing code is required for context, rather than repeat all the code, a ⋮ will be displayed:

www.it-ebooks.info

```
function animate() {
  ⋮
  return new_variable;
}
```

Some lines of code are intended to be entered on one line, but we've had to wrap them because of page constraints. A ➥ indicates a line break that exists for formatting purposes only, and should be ignored:

```
URL.open("http://www.sitepoint.com/blogs/2007/05/28/user-style-she
➥ets-come-of-age/");
```

Tips, Notes, and Warnings

Hey, You!

Tips will give you helpful little pointers.

Ahem, Excuse Me ...

Notes are useful asides that are related—but not critical—to the topic at hand. Think of them as extra tidbits of information.

Make Sure You Always ...

... pay attention to these important points.

Watch Out!

Warnings will highlight any gotchas that are likely to trip you up along the way.

www.it-ebooks.info

Chapter 1

Anatomy of a Website

by Ralph Mason

There are many kinds of websites. They come in lots of sizes, and can be built using a range of tools on a variety of platforms. The aim of this first chapter is to highlight the elements that most websites have in common, and how these elements relate to each other. If you're already familiar with the basics of how websites work, feel free to skip ahead to the next chapters.

Files and Folders

If you've spent any time on a computer, you'll most likely have created, edited, and managed files—such as text documents, images, and videos. You will probably also have created one or more folders, and stored these files inside them.

This, essentially, is what a website is: a bunch of files organized into one or more folders. These files and folders link together in special ways to form the website you view in your web browser.

Hosting

Before examining the anatomy of a website, it's worth noting where a website *lives*.

www.it-ebooks.info

To be available on the Internet, the various components of a website need to reside on a **server**. A server is a computer equipped to make a website available when it's requested through a browser. The server could be your own desktop PC, but more commonly it's a computer dedicated to website hosting, owned by a web hosting company from whom you rent server space.

To be able to serve web pages to the Internet, the server must have special software installed. This includes an operating system and web server software. The most popular operating system for web servers is Linux,[1] though other systems like Windows and Apple's OS X can do the job. The most widely used web server software is Apache.[2]

Amid all the bits and pieces of the server software, there will be a folder in which you place all your website files. This is known as your site's **root folder** or **root directory**. If you rent hosting space from a web hosting company, you'll receive details on how to access this root folder. This is all you really need to know in order to upload your website files and go live online!

Getting Your Site Online

There are various ways to place your website files online. Perhaps the most common way is to upload them with an FTP (file transfer protocol) program, also known as an **FTP client**. You can pay a lot of money for some of these, while others are available for free, such as FileZilla.[3]

Once you have your FTP program, simply enter the address of your root folder, along with a password, and click upload.

The final element you need is a **domain name** (for example, mysite.com). This is like a street address for your site. (It's one thing to have a house, but if you have no street address, no one will be able to find you.) Once you've purchased a domain name from a domain name registrar, there are simple steps for pointing the domain name to your host's computer. That way, when a user types your domain name into a browser, through the magic of the World Wide Web, they will be directed to the website files on your server.

---

[1] http://linux.org
[2] http://apache.org/
[3] http://filezilla-project.org/

www.it-ebooks.info

The Root Folder

Okay, onto the main topic: the anatomy of the website itself. As I mentioned, inside your root folder you place any subfolders and site files (such as text files and image files), just as you store files on your desktop computer. What makes websites special is the way these site assets can be linked, so that they can work together, enabling you to easily jump from one to another from within a browser. How they link together is described further on.

The Index Page

The most important file in your root folder is the index page. On some systems, this may also be called home or default, but index is the most common name. This is usually named something like **index.html** or **index.php**. The index page is the home page of your site, or the page that appears in the browser when a user points their browser to your domain name. When you point to any folder without specifying a page, the server offers up the index page by default.[4]

You could place all the other pages of your website directly inside the root folder too—perhaps giving them names like **contact.html**, **about-us.html**, and so on. But if your site has a lot of pages, this arrangement could become quite unwieldy, so it's best to place other pages within separate folders. Thus, your root folder might contain a whole bunch of subfolders, such as **/contact/**, **/about-us/**, and **/articles/**, in which your Contact, About Us, and Article pages would then reside.

Directory Index

If you have a page like contact.html in the root folder, it will be found at mysite.com/contact.html. If, instead, you place your contact page within a folder called **/contact/**, its URL would be mysite.com/contact/contact.html. But you could also rename this contact page to **index.html**, in which case you could cite its URL as mysite.com/contact/, which is a little nicer looking.

[4] If a folder lacks an index page, visitors will instead see a list of all files in that folder, which is usually undesirable. To prevent this kind of access, place an index page inside each folder—even if the page is just blank.

www.it-ebooks.info

Stylesheets, Scripts, and Images

Apart from web pages, there are other text files that are important to the functioning of your website, even though visitors do not view them directly. These include **stylesheets** and script files. Stylesheets (otherwise known as CSS files, or cascading stylesheets) contain instructions for the visual appearance of your site. Script files—such as JavaScript documents—affect the behavior of your site, such as creating pop-up boxes or powering fancy slideshows.

A good housekeeping practice is to store stylesheets in a separate folder (such as **/css/**), and script files in a scripts folder (such as **/scripts/** or **/js/**, as the mood takes you). It's also common practice to store images in their own folder, which might be named **/images/**, **/img/**, or even **/i/** (if you're a real bandwidth scrooge or disciple of Friar Ockham).

Optional Root Folder Assets

Here are some other items that may be worth placing in your root folder.

Favicon

A **favicon** (short for favorites icon) is a small image file that appears to the left of your site URL in web browsers. It might be a tiny version of your logo, or some other decorative piece. Nowadays, browsers don't require the favicon to live in the root folder, though this was a requirement of older browsers—and frankly, I'm happy to leave it there. Browsers will look for a favicon image in the root folder without you telling them it's there, but you will need to provide a link in your page files if the image is located elsewhere.

A Note on Favicons

A favicon should be square and either sized at 16×16px or 32×32px. Some browsers will accept a file named **favicon.gif** or **favicon.png**, but all will accept **favicon.ico**, so I tend to use that. To create the **.ico** file extension, make sure you use a tool designed for this purpose (such as http://chami.com/html-kit/services/favicon/). Simply renaming the extension to **.ico** will result in incorrect formatting.

www.it-ebooks.info

Robots.txt

You might wish to place a simple text file in your root folder called **robots.txt**. This file is used to instruct web robots (such as Google's indexing spiders) on how to index your site. For example, you may want to instruct search engines not to index certain pages on your site.

It's recommended that you place a **robots.txt** file in your root directory, whether or not you have any instructions for web robots, as they will tend to look for one when they visit your site. It's fine for the **robots.txt** file to be blank. Without a **robots.txt** file at all, you may receive a lot of 404 errors listed in your site statistics (because the spiders are unable to find the file they're looking for), and it's even possible that some search engines won't index your site at all.

Sitemap

You can add a sitemap file to your site, which helps search engines index your pages. (Visit http://sitemaps.org/ for more information on this, or type "sitemaps" into your favorite search engine.) Most commonly, this would be an XML file such as **sitemap.xml**, and it should ideally be located in your root folder.

To let search engines know that your sitemap is available to them, you can place a simple line in your **robots.txt** file, such as:

```
Sitemap: http://www.mysite.com/sitemap.xml
```

See, that **robots.txt** file didn't stay blank for long!

Using .htaccess

If your web host is using Apache to serve up web pages, you can take advantage of **.htaccess** files. An **.htaccess** file is most commonly placed in your root folder, and can be used for a range of purposes, such as redirecting old URLs to new ones.

It can be a little tricky to set up an **.htaccess** file, as filenames with a dot at the front are usually hidden from view on some platforms (Mac and Linux), or tricky to name correctly.[5] The easiest way to work on an **.htaccess** file is through your web host's

[5] If you create a file in Windows and try to name it **.htaccess**, Windows will typically add a **.txt** extension. You can stop this by wrapping double quotes around the name (for example, "**.htaccess**") when you save the file.

www.it-ebooks.info

file manager (if you have access to one). There, you can choose to show hidden files, open your **.htaccess** file, and edit it.

If you want to create an **.htaccess** file through your desktop code editor (I'm thinking software like Dreamweaver here), you have to resort to some trickery.

First, create a file in your root folder and call it **htaccess.txt** or similar; then place the required code inside that file and save the changes. Now upload the **htaccess.txt** file to the server. The next step is to rename this file on the remote server. If you're using an FTP program, you should be able to switch to Remote view, where you see all the files that have been uploaded to the server. Rename the file to **.htaccess** (placing a dot at the front and removing the **.txt** extension).

A Custom Error Page

It's also a good idea to add a custom 404 error page to your site, to avoid the ghastly "404 Not Found" page if a user follows a link to a page that doesn't exist on your site.

You can create a nice page that people will see when they stumble on a bad link. Call this **error.html** or similar, and place it in your root folder. Style the page to your heart's desire (preferably with a friendly apology and a list of pages that visitors might like to go to instead).

Minimum Size

Be aware that the error page may not work unless it has a certain amount of content. To ensure that it works in all browsers, make sure that the file size is at least 4,096 bytes.

Then, in your **.htaccess** file (see, it's already come in handy!), place this code:

```
ErrorDocument 404 /error.html
```

Beyond the Root Folder

As I've mentioned, a lot of your site files are best stored in subfolders rather than in your root folder. Again, this is primarily to be organized, though you also benefit from creating a logical site hierarchy, along with some nice URLs.

www.it-ebooks.info

You can nest folders one inside another as deeply as you like, but don't go overboard. The deeper the pages and files are within your site structure, the harder they may be to find—by your site visitors and search engines.

Include Files

Other than folders for sections of a site (such as **/articles/**), and folders for other site assets such as **/images/**, **/css/**, and **/scripts/**, I like to have what is often called an **includes** folder. This folder contains pieces of content (stored in simple text files) that appear in lots of places across the site, such as the site menu and site footer. If you place the code for items like these separately on every page, you're in for a lot of work if you want to make a change to them. (For example, if you want to add an extra link to your site's existing navigation menu, having to open every page of your site to change the menu is time consuming.) It's much more efficient to store these items in one single file and pull them into your pages where needed.

One way of doing this is to use PHP[6], a scripting language used extensively on the Web. Most web hosts support PHP scripts. If you have some PHP code on your page, your web server will process any instructions in that PHP code before sending the page to a user's browser. A PHP instruction might be to pull in some content from another file, such as the site's navigation menu that you have stored in your **/includes/** folder.

For PHP to work in your pages, it's best to give PHP files a **.php** extension (instead of **.html**). Then all you need to do is create links to the file that you want included on each page. Let's now look at how to do that.

Linking Everything Together

So, files and folders are fine, but how do all these parts link together? After all, links are one of the main features that make the Web so … webby.

Anatomy of a Web Page

A web page is a special kind of text file. In that file, there's both the page content (essentially, the text that visitors will read), along with information that tells browsers about the content. For example, browsers need to be told which text is a heading

[6] http://php.net/

www.it-ebooks.info

and which is a paragraph, where images should go, and what parts of the page will be links.

When a browser downloads a web page, it quickly interprets the nature and relationships of all the elements in the page, and constructs what's known as a **document object model** (or DOM). Browsers follow conventions for constructing this model, providing a useful structure that developers can reference when writing dynamic scripts.

Markup

The instructions that provide this information for browsers are collectively called **markup**. For example, to tell a browser that some text is a top-level heading, wrap it in the `<h1></h1>` tag pair, like so:

```
<h1>This is the Main Heading</h1>
```

The most common markup language used in web pages is **HTML** (hypertext markup language). One important feature of hypertext is the **hyperlink**, which allows users to jump from one place to another. The following `<a></a>` tags tell the browser to create a link to SitePoint:

```
<a href="http://www.sitepoint.com/">Visit SitePoint.com!</a>
```

In your browser, you would just see "Visit SitePoint.com!", but clicking on that text would direct your browser to http://www.sitepoint.com/.

Hence, you can link all the pages and other assets of your website together with these handy hyperlinks.

Page Structure

Whenever you are viewing a web page, you have the option to view that page's **source code**—the page content alongside the markup that a browser uses to render the page.

Near the top of your screen, click **View** > **Page Source** if you're using Firefox or Safari. If you're using Explorer, go to **Page** > **View Source**. And if you're using Chrome or Opera, go to **View** > **Developer** > **View Source**. What you'll see is a distinctive page structure, divided into sections.

www.it-ebooks.info

First, you will usually see a **doctype declaration**. A typical doctype declaration looks similar to this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

The purpose of this declaration is essentially to give the browser information about how to render the page. In crude terms, it tells the browser to assume that you—the web developer—knows what you are doing, and that the page is not constructed with outdated code that must be accounted for.

Next, you should see the opening tag of the actual HTML document, with a **language attribute** to indicate the language in which the page is written:

```
<html lang="en">
```

Most markup tags—such as `<html>`—will prevail until they are closed off with the same tag and a forward slash: `</html>`. The rest of the web page should be contained within the `<html></html>` tag pair.

The code within the `<html></html>` tags is now divided into two main sections: the `<head>` section and the `<body>` section. The `<head>` section (between the `<head></head>` tag pair) contains all sorts of important data and links that remain mostly unseen in the browser, while the `<body>` section (contained between the `<body></body>` tags) contains the content that is viewed by visitors to the site. So the essential structure of a standard HTML page looks like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
    <head>
        … important data (mostly) not seen by visitors
    </head>
    <body>
```

www.it-ebooks.info

```
        … page content viewed by visitors
    </body>
</html>
```

The <head> Section

The <head> section always sits above the <body> section of a web page. Here are some of the items it commonly contains.

Meta Elements

The <head> section typically contains a number of meta elements (or elements that identify properties of the document):

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

This line provides some useful information to the browser. It confirms that the page is an HTML document, and specifies the document's character encoding (which is mainly useful when you are viewing the document offline).[7] Normally, the actual character encoding is determined by settings within the web server; this won't apply when viewing HTML documents on your computer, outside of a server environment.

The meta **description** and **keywords** can help search engines to understand the content of the page:

```
<meta content="SitePoint.com - the best site for web developers."
  name="description">
<meta content="web design, accessibility, standards compliant"
  name="keywords">
```

Title

The title element is mandatory, and provides a page title that browsers display to site visitors.

```
<title>SitePoint : Become a Better Web Developer</title>
```

[7] There are many different character encodings, but UTF-8 is usually the best one to choose, as it helps your browser to display a wider range of characters and symbols than any other encoding.

www.it-ebooks.info

Styles and Scripts

A stylesheet link connects the page to the CSS stylesheet, which tells the browser how to style the page—from specifying the layout of the various elements, to dictating such factors as colors, fonts, and font sizes. It is common to have multiple stylesheet links:

```
<link type="text/css" media="all" rel="stylesheet"
  href="/css/site.css">
```

A script link connects the page with a script that directs the browser to initiate certain dynamic page behaviors. As with stylesheet links, it's common to have several script links in the head of the document:

```
<script type="text/javascript" src="/js/whiz.js"></script>
```

Scripts at the Bottom!

Many developers find it more practical to place script links just before the closing `</body>` tag. This is so that the page loads visually in the browser before dynamic scripts are downloaded—as the browser downloads the page from top to bottom, including linked files. This practice also helps to ensure that page elements have loaded in the browser before the scripts that affect them begin to operate. A common example of scripts at the end of the page is the Analytics code provided by Google as part of its webmaster tools (which measure such aspects as traffic to and from your site).

Rather than link to external stylesheets and script files, you have the option to embed styles and scripts in the page itself. However, only do this if those styles and scripts will be used on this page alone—as other pages won't be able to access them here. If the scripts and styles are used by other pages, it's much more efficient to store them in one place that all pages can access.

To embed CSS styles on the page itself, you place them between the `<style>` tag pair:

```
<style type="text/css" media="all">
  … styles go here
</style>
```

www.it-ebooks.info

To embed scripts on the page, place them between the <script> tag pair:

```
<script type="text/javascript" src="">
  … scripts go here
</script>
```

Favicon

The favicon link tells the browser where to find a favicon image. It's optional if the favicon resides in the site's root folder, but is essential if the favicon resides elsewhere:

```
<link rel="icon" href="/favicon.ico"
  type="image/vnd.microsoft.icon">
```

Similar to favicons are Apple touch icons that represent bookmarked sites on iPods, iPhones, and iPads. You can specify your own icon with this link:

```
<link rel="apple-touch-icon" href="/custom-icon-name.png">
```

There are many more meta and other links that can reside in the <head> of the document, such as copyright information and RSS links, but this is a sufficient introduction for current purposes.

The <body> Section

Within the <body></body> tags goes the content of the web page—including text, forms, links to images, and so on. I'll avoid going into too much detail here, as doing so would essentially constitute a lesson on HTML markup; but a few points are worth mentioning.

Links

Links to other pages—perhaps the "bread and butter" of the Web—can be placed pretty much anywhere within the body of your web page, such as within a paragraph:

```
<p>This is a <a href="article1.html">link to an article page</a>.</p>
```

Or even around an image:

www.it-ebooks.info

```
<a href="/articles/article1.html">
  <img src="/images/entice.jpg"
       alt="An enticing image to make you click here">
</a>
```

Images

Images that you have stored in your **/images/** folder can be linked to (or pulled into the page) like so:

```
<img src="/images/myimage.jpg" alt="A beautiful image">
```

Includes

As mentioned, include files hold content you want placed on multiple pages. They can be pulled from an **/includes/** folder like so:

```
<?php include $_SERVER["DOCUMENT_ROOT"] . "/includes/menu.html"; ?>
```

Internal Links

You can also link from one part of a page to another within the same page. For example, you might have a link near the top of your page pointing to related content further in:

```
<p><a href="#more">Read more, further down the page!</a></p>
```

The word "more" in this instance is known as a **fragment identifier**. The hash symbol (#) followed by this unique identifier (which can be almost any name you like, but it must be unique[8]) provides a way to link to a specific element with an `id` matching the identifier. So if, for example, you have a paragraph on the page like this:

```
<p id="more">There is a lot more information here…</p>
```

The browser will jump to this paragraph when you click on the "Read more" link.

[8] You can include letters, digits, dashes, and underscores, but no spaces. Try to choose meaningful identifier, so that you don't come back later and wonder what that ID is referring to.

www.it-ebooks.info

Note that you can jump to this same paragraph from a different page (or from another website) by citing the full URL to the page with the unique identifier appended:

```
http://www.mysite.com/articles/article1.html#more
```

A Note on File Paths

In all the preceding link examples (such as `src="/images/entice.jpg"`), I've placed a forward slash at the start of the file path. This is a handy way of linking to files, but it's far from the only way. Here are three standard ways this path might be written:

```
http://www.mysite.com/images/entice.jpg
```

```
/images/entice.jpg
```

```
images/entice.jpg
```

The first is called an **absolute path**. This is the full URL that could be cited from any web page, anywhere on the Web, and still take you to the **entice.jpg** image.

The second and third ways are called **relative paths**. They can only be used from within the site itself.

The second link (/images/entice.jpg) is *relative to the site's root folder*—by virtue of the forward slash at the start. Thus, it's saying to the browser: "Look for a folder called **/images/** in the root folder, and therein find the **entice.jpg** image."

The third link (images/entice.jpg), rather than being relative to the site's root folder, is *relative to the page itself* (that is, the page on which the link appears). It's saying to the browser: "Look for a folder called **/images/** in the folder where this page resides, and inside that, find the image called **entice.jpg**." On another page within the site, a link of this kind might appear like this:

```
../images/entice.jpg
```

This would be saying to the browser "Go to the folder that contains the folder this page lives in, and therein find a folder called **/images/**, in which you'll find the **entice.jpg** image." So two dots (periods) followed by a forward slash means "go

www.it-ebooks.info

up one folder." On sites with deeply nested pages, it's not uncommon to see file paths with many dots and dashes—for example, ../../../images/entice.jpg.

Each of these three methods of linking to files within the same site has its pros and cons. The first—which uses the absolute path—is very reliable. It uses the full, unique path to a file that no other file on the Web shares. Some people argue that absolute URLs also confer some benefits in terms of search engine optimization. On the downside, however, an absolute path includes more information than is needed to find an internal site file, and will increase (albeit negligibly) a page's file size.

What's more, using the full URL is quite inefficient. You're telling the browser to call out across the entire World Wide Web for a domain's location, when it already knows where your site is located. It's as if a visitor to your home asked where the bathroom was, and you answered with your entire address (country, state, street address, and zip code) before ending with "down the hall, first door on the left."

One further consideration is that, if you ever move your site to another domain, you'll have to rewrite all the absolute file paths.

The second, root-relative path is very handy, because you can use the one same link from anywhere within a site, and it will always point to the same location. However, it only works in a server environment—such as on your web host's server, or in a server environment that you have set up on your computer.[9] If you're just testing simple HTML files on your desktop, such links won't work.

Page-relative links will, by contrast, work nicely when you're testing pages on your desktop computer—whether or not your pages exist within a server environment. And of course, there will be no need to change anything when you transfer these files to the Web. The drawback is that you have to be particularly careful when constructing these file paths. The paths to a single file may differ for each page, and it's easy to get them wrong. Despite having worked with them for some time, I still have to think hard about how many levels up or down I need to go, and how many dots and slashes I will need, and so on. To be honest, it's usually too much work for my poor brain, so I normally work within a server environment and use root-relative paths.

[9] A particularly easy and quick way to set up a simple server environment on your desktop—complete with Apache, PHP, and a database—is to install MAMP [http://www.mamp.info/] for Mac or WAMP [http://wampserver.com/en/] for Windows.

www.it-ebooks.info

Beyond Static Sites

What I've described is typical for **static** sites; these are sites that stay the same until you manually edit them.

Enter the CMS

Many sites, however, are more dynamic than this. There are lots of software packages—both free and commercial—that are designed to manage your site content in more sophisticated ways. Often referred to as a **CMS** (content management system), this software package creates new pages on the fly, automatically establishes complex relationships between pieces of content, and stores the site's content in a database.

There are many kinds of CMS, and they are often built for a particular purpose. That purpose might be to power a medium to large business site, a blog, an ecommerce site, a web forum, or a wiki (such as Wikipedia).

If you want to use CMS software to manage your site, you'd normally upload a package of files and folders to your root folder, then log in to a control panel through which you manage your site.

From this point, most of the organization of your site content is handled by the CMS. You can happily enter content—like text and images—through a clean, code-free interface, without having to worry about where the various bits of content is stored.

Even in this scenario, you still have a root folder to manage, and you'll likely want to make use of the root folder assets described above. Essentially, the principle is the same: your root folder will contain a collection of files and subfolders, even if you don't understand what most of them are for!

The Database

It's worth saying a few words about the database, even if you never go near it in practice.[10]

A database is a powerful repository of site content in a dynamic website. The data held by a database is managed by specially designed software (sometimes called a

[10] Well, if your site uses a database, you should at least make backup copies of it at regular intervals.

www.it-ebooks.info

database management system). Perhaps the most popular of these is MySQL.[11] This software is typically installed on your web server—normally by your web host. If you have employed a CMS to manage your site's content, the CMS will store content in the database, and pull it back out again when required—without you having to worry about how. All of this database information is stored within protected folders on your server, so that no malicious visitors can tamper with your precious data.

Conclusion

So, that's my basic take on the anatomy of a website. I hope this quick tour has not left you with too many unanswered questions—although that wouldn't necessarily be a bad thing. If you're bubbling with questions, head over to the SitePoint forums[12] where there are friendly people ready to answer your questions.

Among the excellent resources available at SitePoint, I highly recommend the SitePoint Reference,[13] a treasure trove of information on HTML, CSS, and JavaScript. Each of the forums has a collection of sticky threads with links to further resources and answers to commonly asked questions. And of course, SitePoint has a superb collection of books that provide in-depth guidance on all aspects of web development.

Best wishes with your explorations of all things Web!

[11] http://mysql.com/
[12] http://sitepoint.com/forums/
[13] http://reference.sitepoint.com/

www.it-ebooks.info

www.it-ebooks.info

Chapter 2

Designing in the Dark

by Alex Dawson

Whilst unsung, the average web designer (or developer) is a hero of the digital world. Though this may sound rather egotistical coming from a web professional, the task of building sites and experiences involves much more than throwing code into a web browser. We provide experiences and services that many people rely upon in their daily lives.

Because people the world over are diverse in culture and needs, a common problem designers aim to solve is the mystery of an "average user." Without knowing what users require, we can't guarantee that the experience matches expectations. In order to avoid designing in the dark, we shall now examine this tricky task and aim to reduce the guesswork that can go on.

The Case of the Unknown Visitor

Our differences define us, making us unique and special. Accounting for matters of taste, relevance, and usefulness within design is a complex task. Most people in our position rarely gain the opportunity to speak to our web visitors to any great

www.it-ebooks.info

extent. That being said, there are certain actions we can take to make the unknown visitor a real person.

Setting the Scene

The idea that "everyone" can access the Web or has JavaScript turned on is proof of the assumptions made of our industry, serving to cloak the unknown visitor in a veil of mystery! Before we come to understand who a visitor is, it's important to acknowledge that your audience consists of real people, with real thoughts and feelings.

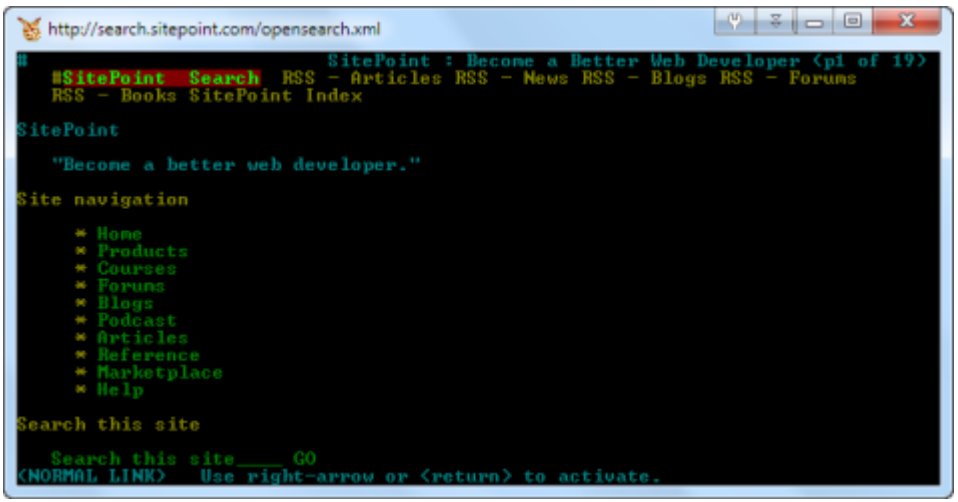

Figure 2.1. Who is the end user? Perhaps it's a Lynx user with no scripting

Importantly, these people have the power to make decisions for themselves. Any preconceived notions you may have as to what your users have installed or are using, or their level of ability, should be thrown out the window immediately.

Who is the visitor?

So, if we don't know who the visitors will be, how can we depict them as real people? By deducing a few factors about the type of people who *may* visit your site, we can at least have a rough understanding of the variables involved. Before you snoop around to gauge a visitor's needs and expectations, you must consider these factors.

www.it-ebooks.info

Figure 2.2. Some sites like Wikipedia serve multiple demographics

To calculate the type of visitors that exist, you just need to think of the ways in which people differ. There are many ways to define an individual; a few are shown below. The way to determine such groups is to analyze who you want to target, and who do you think will find your services appealing.

Examples of variables which may define an audience:

- gender (male or female)
- career (job relevancy or training skills)
- culture (country, traditions, religion, or ethics)
- education (literacy level or technical knowledge)
- age (child, adolescent, adult, or silver surfer!)
- ability (impairments or restrictive issues)

You're the Detective

Picture, if you will, a figure like Sherlock Holmes. Like all great detectives, it's his job to follow a series of clues to solve the mystery and unmask the villain. Okay, our visitors aren't the bad guys here (except perhaps a few of those clients we encounter from time to time), but methodical thinking and a thirst for knowledge drives us to calculate who we're designing for.

www.it-ebooks.info

A Challenge Ahead

Within this context, I put it to you, dear reader, that while considering how you'll design a page you put a few of those detective skills to work. Just like a good general would not go to war without a battle plan, in your quest for a fantastic site, you must know what you're dealing with and not charge into the unknown. Much of this philosophy is built on common sense.

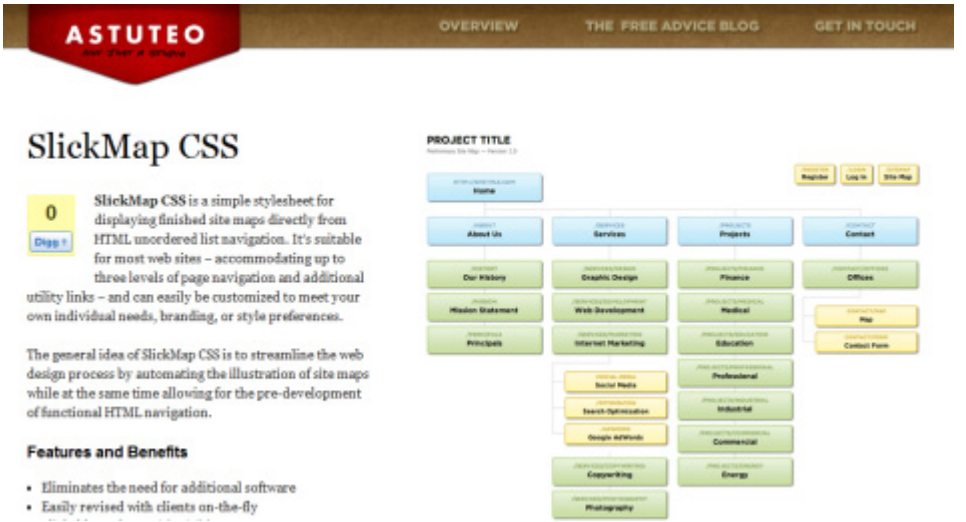
ASTUTEO
OVERVIEW
THE FREE ADVICE BLOG
GET IN TOUCH
SlickMap CSS
SlickMap CSS is a simple stylesheet for displaying finished site maps directly from HTML unordered list navigation. It's suitable for most web sites – accommodating up to three levels of page navigation and additional utility links – and can easily be customized to meet your own individual needs, branding, or style preferences.
The general idea of SlickMap CSS is to streamline the web design process by automating the illustration of site maps while at the same time allowing for the pre-development of functional HTML navigation.
Features and Benefits
Eliminates the need for additional software
Easily revised with clients on-the-fly
PROJECT TITLE

Figure 2.3. Sitemaps may seem unnecessary, but they reveal page relationships

Perhaps you find this daunting, but conducting research and knowing how a business or site may be affected is essential if you want to cater to the right audience. Even the most well-written content or functional service will be useless if there's no audience. As the site's detective, you must have a plan and be prepared.

Your web design battle plan requires you to know:

- your business model
- the skills you require
- your audience's needs
- how to take action

www.it-ebooks.info

Why It Really Matters

Ultimately, there's nothing better than doing a job well and knowing that your message reached its intended visitors. Because your audience justifies the site's existence, we need to place an increased level of effort into meeting their needs. If carried out properly, a website is more likely to succeed, and you'll enjoy a positive experience.

Figure 2.4. SitePoint succeeds because it knows how to draw an audience

Figure 2.5. You don't want to end up with a website like this ... do you?

www.it-ebooks.info

Clues, Evidence, and Theories

So visitors are unique, and determining their needs is hard work. How are we meant to work out what these people need, want, and expect? We take the concept of being digital detectives to the next level—going beyond objective deduction to look for clues, find witnesses, and separate good ideas from the bad … exciting!

Great Expectations

Your visitors are going to have a range of expectations of how your service is run. Some of these will be quite understandable (such as having an easy-to-use navigation menu), while others may be impossible to implement. When you're unable to meet such expectations (for whatever reason), it's important that you respond to such requests and justify why.

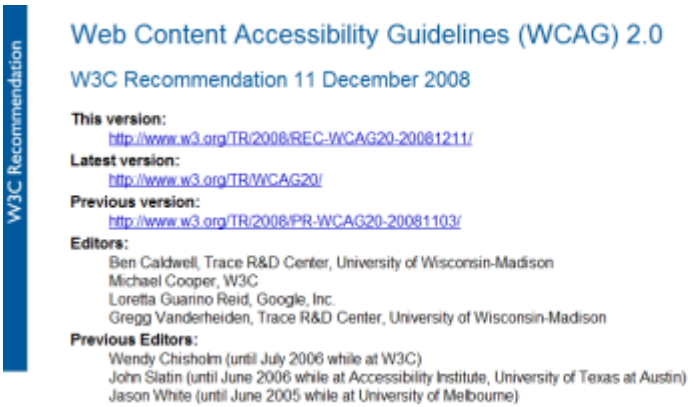
W3C Recommendation
Web Content Accessibility Guidelines (WCAG) 2.0
W3C Recommendation 11 December 2008
This version:
http://www.w3.org/TR/2008/REC-WCAG20-20081211/
Latest version:
http://www.w3.org/TR/WCAG20/
Previous version:
http://www.w3.org/TR/2008/PR-WCAG20-20081103/
Editors:
Ben Caldwell, Trace R&D Center, University of Wisconsin-Madison
Michael Cooper, W3C
Loretta Guarino Reid, Google, Inc.
Gregg Vanderheiden, Trace R&D Center, University of Wisconsin-Madison
Previous Editors:
Wendy Chisholm (until July 2006 while at W3C)
John Slatin (until June 2006 while at Accessibility Institute, University of Texas at Austin)
Jason White (until June 2005 while at University of Melbourne)

Figure 2.6. Visitors expect an accessible site, so follow WCAG

Any facilitator to improving the website's usefulness and reducing any barriers to entry are a good idea, especially if you're trying to compete with similar services. With this in mind, you need to work out what's required per project. While the site you're working is yet to exist, there are ways to get some solid research.

www.it-ebooks.info

Silverback has been invaluable as we make tweaks/test the new mN site. Thank you, Clearleft!
so says Brendan Dawes, interaction designer | Next testimonial »
Clearleft presents
Silverback 2.0
Guerrilla usability testing software for designers and developers
Capture screen activity
Video the tester's face
Record the tester's voice
Add chapter markers on-the-fly
Control recording with the remote
Export to Quicktime
NEW FEATURES
Features in 2.0 include
Preview
Batch Export

Figure 2.7. Tools exist to help you manage this, but the hard work is up to you

Innovative Ideas

When scoping sources for innovation, you need to look beyond what's in front of you. Ways to gather ideas may include visiting a competitor's site to see what they offer clients, asking people who want such a service what they look for, or researching how this service could be built.

Here are some practical suggestions for website innovations:

- Examine inspirational sites for specific layout ideas.
- Use frameworks to aid the agility of certain effects.
- Research what your visitors need—usability is critical.
- Examine how your competitors implement elements.
- Fix problems when they occur.

Gathering evidence to justify a service ensures that you have a product that people want (saving development time on dud projects), and gives you an idea of the type of audience you may attract. While this is straightforward, it's worth remembering that even when your site is complete, you (or the client) need to keep innovating and evolving the project to maintain its longevity.

www.it-ebooks.info

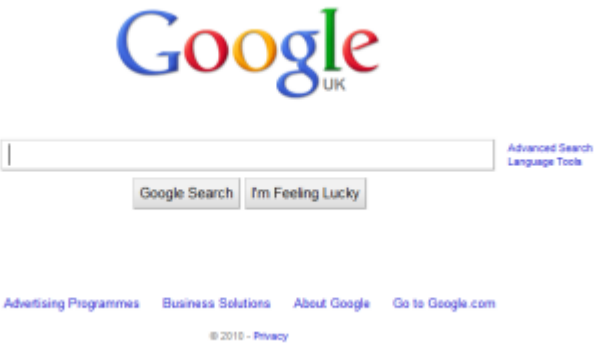

Figure 2.8. Google made its fortune through continual innovation

Building Visitor Solutions

Enticing visitors to our websites often comes down to our knowledge of a group's identity. For every implementation and idea that exists, the solution serving the most users possible can be offered online.

Examine the Problems

Everything that appears in a design should aim to solve some kind of problem. Whether it's the colors we choose for readability, the content we write to assist our visitors (and gauge their interest), or the design conventions we use to maximize website usability. The goal of a web designer is to create experiences that offer solutions, perhaps even a fresh perspective.

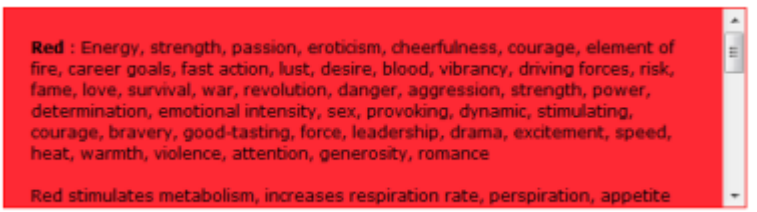
Red : Energy, strength, passion, eroticism, cheerfulness, courage, element of fire, career goals, fast action, lust, desire, blood, vibrancy, driving forces, risk, fame, love, survival, war, revolution, danger, aggression, strength, power, determination, emotional intensity, sex, provoking, dynamic, stimulating, courage, bravery, good-tasting, force, leadership, drama, excitement, speed, heat, warmth, violence, attention, generosity, romance
Red stimulates metabolism, increases respiration rate, perspiration, appetite

Figure 2.9. Color plays a psychological role that can influence emotions

As you design your site, refer to any research you've undertaken and ask yourself: What would my visitors want? Who are my visitors? Why are they here? How can I make things easier for them? Placing yourself in the shoes of others or undertaking

www.it-ebooks.info

usability studies (to analyze the results of page interactions) can present you with a benchmark, or a possible way forward.

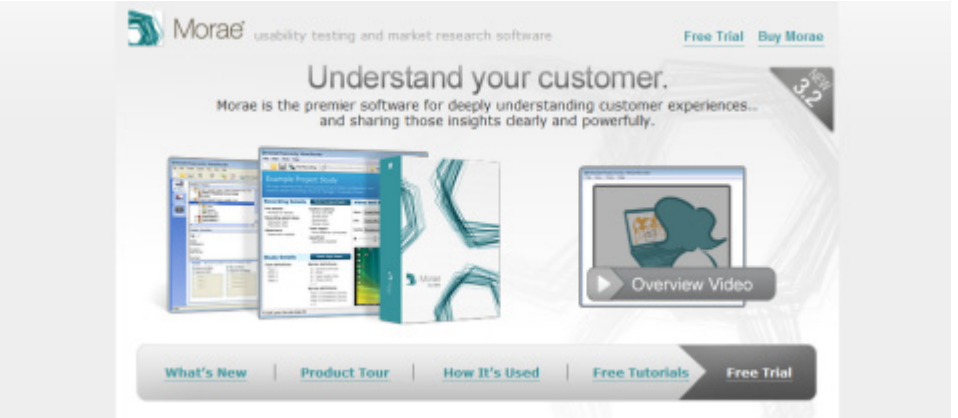

Figure 2.10. Usability testing software can help you know your customers

Implement Experiences

Witnessing an implementation work in the way a visitor expects it to is the primary goal of conducting such investigations; you're also able to gain some insight into how it's expected to function. Innovation is worthy, but you need to ensure that your choices are objective and based fundamentally on usability.

Figure 2.11. Users expect a logo to appear in the top-left corner

On that note, it's worth stating that you can't please everyone. The ideal user experience will differ depending on who's being asked, and the research you undertake may uncover a range of opinions. For the best route to take, ensure a site is primarily accessible, and then account for functionality when required; you want to avoid a bloated page.

When considering the user experience, web designers need to account for:

- code semantics
- web accessibility

www.it-ebooks.info

- usability issues
- content value
- service standards

Learning from Experience

You can't expect to know what's good for your audience right off the bat—we're not a bunch of mystics after all. What you can do, however, is give the research process a go and make sensible choices based on the circumstances. If you've yet to communicate with your audience, now is the time to try!

Trial and Error

It's impossible to get things right all the time, and it's a fact of life that even the simplest piece of software will have a bug somewhere. While a site is not a piece of software, the idea of error occurrence is as potent as it ever. Some designers fear coding due to the complexity of the idea, that they might make a mistake … but that shouldn't hold you back from trying.

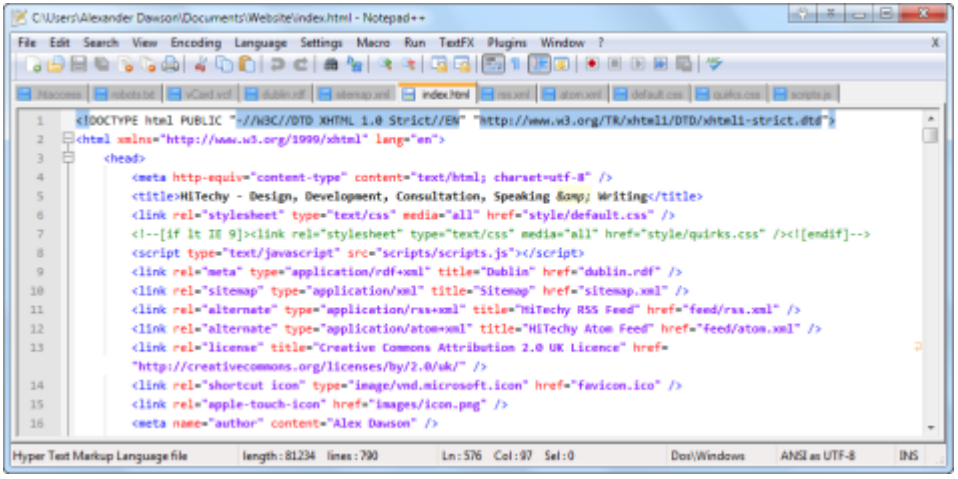

Figure 2.12. Playing in a code editor can be lots of fun, as is trying out new code

Getting to know your visitors is a lot like coding for different platforms. Some people you'll hit it off with and understand their niche requirements straightaway (like Opera); others will have a wide range of needs while still being patient and understanding (like Firefox); and some will just drive you insane (like IE6). Innovating and working with these people is all part of the job you do.

www.it-ebooks.info

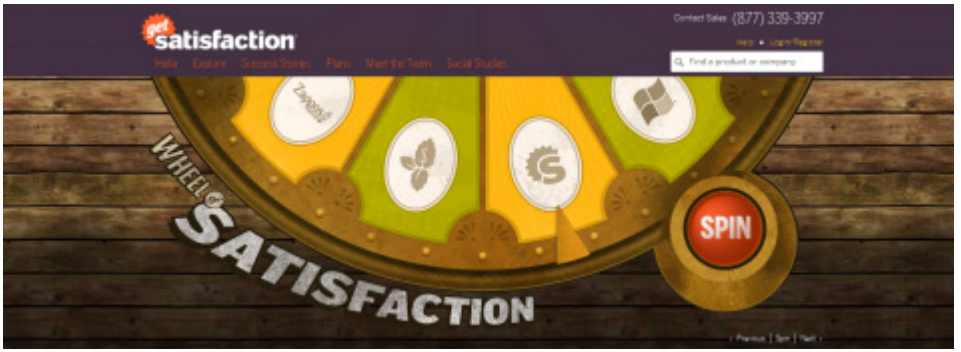

Figure 2.13. Sites like Get Satisfaction offer the means to talk with visitors

Convention Coding

By studying user behavior, usability, and accessibility, we can see patterns and conventions with how people adapt to functionality (like navigation layouts). Knowing your visitors and their needs will allow you to produce a design that adopts conventions while enabling you to factor in browsing habits.

Types of user needs in accessibility include:

- physical
- intellectual
- emotional
- social
- mechanical

The worst thing any website can do is ignore its users. Unfortunately, this occurs far too often. When building a website, it's usually the boss's idea of what is good and bad that results in the finished site; the user's needs, which should be paramount, get degraded to the class of secondhand citizens. Using conventions can help avoid this.

www.it-ebooks.info

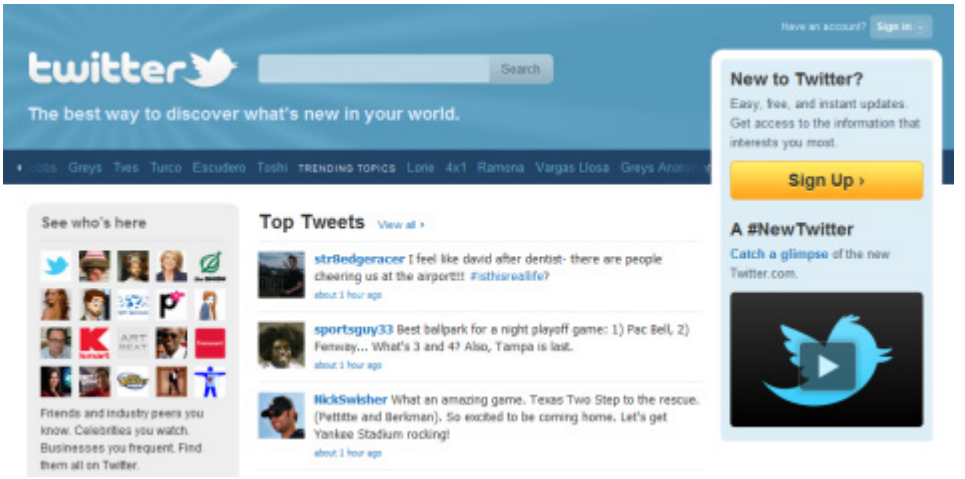
twitter
Search
The best way to discover what's new in your world.
TRENDING TOPICS
New to Twitter?
Easy, free, and instant updates. Get access to the information that interests you most.
Sign Up ›
A #NewTwitter
Catch a glimpse of the new Twitter.com.
See who's here
Friends and industry peers you know. Celebrities you watch. Businesses you frequent. Find them all on Twitter.
Top Tweets
str8edgeracer I feel like david after dentist- there are people cheering us at the airport!!! #isthisreallife?
sportsguy33 Best ballpark for a night playoff game: 1) Pac Bell, 2) Fenway... What's 3 and 4? Also, Tampa is last.
RickSwisher What an amazing game. Texas Two Step to the rescue. (Pettitte and Berkman). So excited to be coming home. Let's get Yankee Stadium rocking!

Figure 2.14. Social networks like Twitter can help with damage control

Seeing the Light

Focusing on the user is ideal, because 99.9% of the time, that's who you're trying to impress. Without this focus, time, money, and effort may be squandered. So many services today compete in a heavily crowded field; if web designers focus more on their visitors, they'll have a distinct advantage.

As a user experience fan, my views are based upon the idea that content is more important than design (substance over style), and that the user is more important than content—that is, with no one to read it, the content's value diminishes. As such, playing the role of detective and spending time to your visitor's needs (and identity) will yield better quality results.

www.it-ebooks.info

Chapter 3

Everything Must Go!

by Alex Dawson

So, you've finished that wonderful website layout and everything works. That is, until you use an old version of Internet Explorer, a mobile device, or some long forsaken piece of technology that's unable to support the modern standards that many of us have become accustomed to. While it's great that we push the limits of our technologies to attain better designs, we still need to consider compatibility.

I'm not one of those hardcore pixel-pushers who force perfection, and I'm not a code monkey that hacks his code to make it look stunning in every iteration or situation. However, I do believe that we (as professionals) need to pay closer attention to why our sites are less than bulletproof, and address what happens when things become disabled. After all, you don't want your visitors to suffer, right?

The Hypertext Apocalypse

Times have changed from back in the era when HTML mostly comprised a few tags and no stylistic enhancements were required. These days, we depend so heavily on technologies like CSS, JavaScript, and Flash that little thought goes into the im-

www.it-ebooks.info

plications for the end user. What happens if such a technology is disabled? Most sites would collapse—Armageddon style!

Crippling Decisions

The decision as to whether to support technologies like Flash may have been arrived at with the release of the iDevices (that is, the iPod, iPhone, and iPad) and their disregard for this technology. However, for the very features we use daily, like images, CSS, and scripting, their loss and how it should be handled is an important consideration for any designer.

Figure 3.1. The iPhone may be functional, but it won't work with Flash

While it's worth declaring at this point that I, like many others, am firmly in favor of sensible code usage, there are implications when one of the many implementable enhancements we use regularly is crippled. As to why people disable certain technologies, it may be a matter of security or choice, or even that the browsers in use are below par.

www.it-ebooks.info

Figure 3.2. Without Flash, this site is doomed—it has no fallback ability

Accessibility Matters

Arguably, there are many out there who would consider the idea of having a site without scripting insane; however, the user is of prime importance, so accessibility is a very justifiable concern. Having a website comes with the responsibility of ensuring it works well for as many people as possible … no matter their disadvantages.

Accessibility also has legal implications:

- ADA (*Americans with Disabilities Act*) | Related: Section 508 / WCAG
- DDA (*Disability Discrimination Act*) | Related: PAS 78 / BITV / Stanca Act

Within the construct of accessibility is a principle named "gracefully degrading code." In essence, the notion behind it is that if some part can be disabled or is unavailable, the code must have a fallback to ensure the information's integrity remains. Unobtrusive scripting that has an alternative solution is a clear example of such sensible decision-making.

www.it-ebooks.info

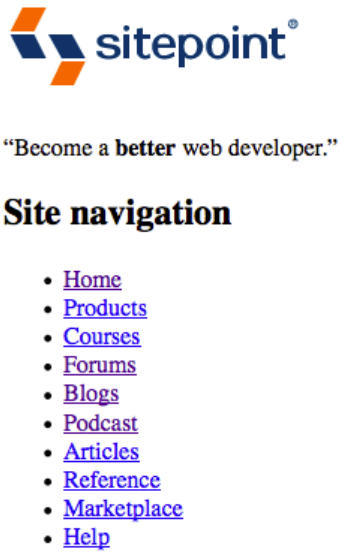

Figure 3.3. Behold a naked SitePoint with no CSS running on the page

Senseless Dependence

There are many ways in which technologies can be made unavailable to us. It's also true that sometimes we depend a little too much on certain technologies in order for sites to work. With the Web still evolving and awesome new technologies arriving all the time, it's worth having a graceful fallback mechanism to give even the most basic of browsers the content they seek.

Disabled Mediums

Some of the disabled mediums on the Web are the result of being able to turn off functionality, be it directly within the browser or through the use of an extension. This also extends to conditional (browser-specific) code, which can always be fooled into not being activated. Sometimes, this is through the user's choice, but it could also be an imposed restriction on the machine being used.

Some of the mediums that can be disabled include:

- Images (foreground and background)
- Frames and iframe content navigation
- CSS stylesheets (individually or totally)

www.it-ebooks.info

- JavaScript (and certain code by default)
- VBScript (within Internet Explorer)
- Plugin content

Because of this ability to restrict (even for the user's own protection), it stands to reason that we should never discriminate against our users, or blame them for requiring a situation that you failed to account for. Disabled features is part of the web game—even if it's the result of a school disabling Flash to stop students from playing Desktop Tower Defense!

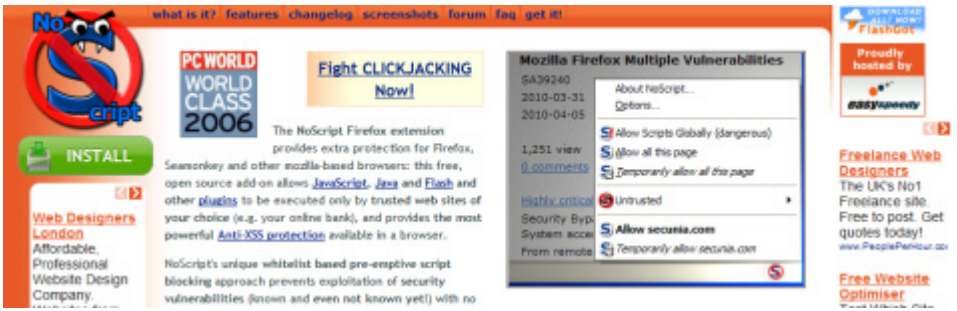
what is it? features changelog screenshots forum faq get it!
INSTALL
PC WORLD
WORLD CLASS
2006
Fight CLICKJACKING
Now!
The NoScript Firefox extension provides extra protection for Firefox, Seamonkey and other mozilla-based browsers: this free, open source add-on allows JavaScript, Java and Flash and other plugins to be executed only by trusted web sites of your choice (e.g. your online bank), and provides the most powerful Anti-XSS protection available in a browser.
NoScript's unique whitelist based pre-emptive script blocking approach prevents exploitation of security vulnerabilities (known and even not known yet!) with no
Mozilla Firefox Multiple Vulnerabilities
SA39240
2010-03-31
2010-04-05
About NoScript...
Options...
1,251 view
0 comments
Allow Scripts Globally (dangerous)
Allow all this page
Temporarily allow all this page
Untrusted
Allow secunia.com
Temporarily allow secunia.com
Proudly hosted by
Freelance Web Designers
The UK's No1 Freelance site. Free to post. Get quotes today!
Free Website Optimiser
Web Designers London
Affordable, Professional Website Design Company

Figure 3.4. Even if a user has NoScript installed, don't shoot the messenger

The Curse of Plugins

When content is rendered poorly on the page, the most common offender is the plugin. Adobe Flash has seen a huge level of support from the design community over the years and it's served a noble purpose of providing functionality that would otherwise have not existed. The only problem is that it requires the extension to be installed in a browser!

A few well-known plugin technologies are:

- Adobe Flash
- Microsoft Silverlight
- Microsoft ActiveX
- Java Runtimes
- Adobe Reader (PDF)
- .NET Framework
- Adobe Shockwave

www.it-ebooks.info

Taking into account the high level of plugin usage, you could be forgiven for thinking that this wasn't a real issue. But the Apple-Flash situation has showcased the huge industry-wide impact such nonstandardized features can have if a manufacturer chooses to ignore it. With additional factors such as security implications and the ease of disabling, a site with no other content may well be doomed.

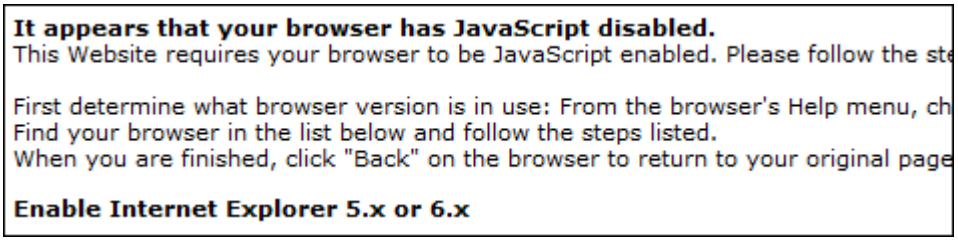
It appears that your browser has JavaScript disabled.
This Website requires your browser to be JavaScript enabled. Please follow the ste
First determine what browser version is in use: From the browser's Help menu, ch
Find your browser in the list below and follow the steps listed.
When you are finished, click "Back" on the browser to return to your original page
Enable Internet Explorer 5.x or 6.x

Figure 3.5. Telling people to enable a plugin is unproductive

Accessible Assassinations

Up to this stage, we've discussed how certain functions can be turned off if they behave poorly. So what can we do about it? Well, you could choose to ignore it (leaving your site inaccessible to the many), take action (provide fallbacks), or do what's often done for IE—hack your way to equality. I'd recommend the second option.

Spraying Weed Killer

To ensure your website gracefully degrades, follow this methodology: go through a list of the functions that can be disabled, turn them off, see what happens, and fix any code where appropriate. It may seem surprisingly simple, and you're probably wondering where the catch is … but there isn't one!

What to test for:

- images and "containers" (plugins) have alternative content
- websites work when CSS, JavaScript, (or both) are turned off

To produce something as complex as a Flash-powered site with an HTML fallback in the background will probably require a fair bit of work. For a smaller job—such as having alternative content in an alt attribute—you should already know that

www.it-ebooks.info

providing the data serves a useful purpose. Every alternative has its own route to success.

Figure 3.6. Without alternative text content, disabled images will drop dead

Content Matters!

If there's one point that can be drawn from the need to gracefully degrade (for those stone-aged browsers), it's that content is king. Arguably, the most graceful degrade you can achieve within a site is that the HTML structure and content remains. If you've ever tested your site using a copy of the text browser Lynx, you'll see how different it can be.

A LIST APART
FINDINGS FROM THE SURVEY FOR PEOPLE WHO MAKE WEBSITES 2009
Introduction
I Age
II Gender
III Ethnicity
IV Job title
V Geographic region
VI Top 20 responding countries
VII Education
VIII Relevance of education
IX Excited by field
X Have a personal site/blog
XI Time personal site/blog online
XII Type of organization
Findings from the A LIST APART Survey, 2009
Once again, A List Apart and you have teamed up to shed light on precisely who creates websites. Where do we live? What kind of work do we do? What are our job titles? How well or how poorly are we paid? How satisfied are we, and where do we see ourselves going?
Once again, we present our findings on the web, with XHTML table data converted to beauteous charts care of CSS, Jason Santa Maria, and Eric Meyer. Others who worked on these findings include editor Krista Stevens and publisher Jeffrey Zeldman.
Analyses contained in this report should be considered primarily descriptive; no attempt was made to assess causality among survey variables. In plain English, be careful not to extrapolate the observations that follow into predictive or causal relationships.

Figure 3.7. Content is a valuable asset, so ensure it's both visible and accessible

Because content is king, ensuring it's always visible is critical to your website's success. You're likely to have every situation under the sun thrown at your resources from your visitors, so if nothing in your site functions to the extent that it should, having alternatives will at least soften the fall from grace.

www.it-ebooks.info

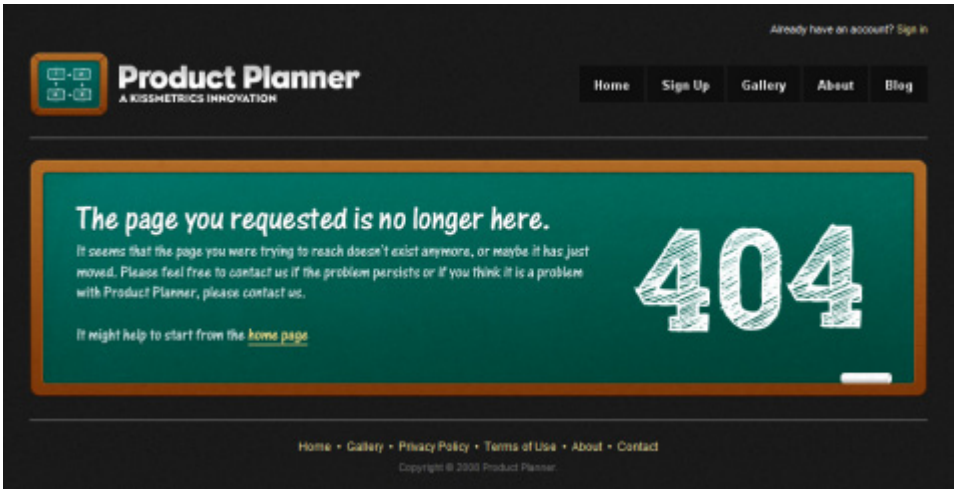
Already have an account? Sign in
Product Planner
A KISSMETRICS INNOVATION
Home
Sign Up
Gallery
About
Blog
The page you requested is no longer here.
It seems that the page you were trying to reach doesn't exist anymore, or maybe it has just moved. Please feel free to contact us if the problem persists or if you think it is a problem with Product Planner, please contact us.
It might help to start from the home page
404
Home • Gallery • Privacy Policy • Terms of Use • About • Contact

Figure 3.8. Even a custom 404 page can degrade an error with ease

Progressive Enhancement

While ensuring that your existing material gracefully degrades is paramount to the design process (along with the accessibility of your content), the flip side of the coin—progressive enhancement—works in reverse. If you have an existing site that needs an accessibility boost, graceful degrading will help you patch your wounds. For new sites, enhancement is a cleaner option.

Starting from Scratch

With a new site design, having to disable regularly can be a pain; however, the requirement to ensure graceful degradation is not lost at this stage. The idea behind progressive enhancement is very simple. You start by marking up your content with HTML, then the CSS, then JavaScript, and finally any additional technologies you want to use. So far, it's fairly straightforward.

www.it-ebooks.info

Zen Garden
The Beauty of CSS Design
A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.
Download the sample html file and css file
The Road to Enlightenment
Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.
Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.
The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.
So What is This About?
There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.
select a design:
Under the Seat by Eric Steltz
Make 'em Proud by Michael McAghon and Scotty Reifsnyder
Orchid Beauty by Kevin Addison
Oceanscape by Justin Gray
CSS Co., Ltd. by Benjamin Klemm
Sakura by Tatsuya Uchida
Kyoto Forest by John Politowski

Figure 3.9. The CSS Zen Garden is a classic example of progressive enhancement

The reason for building the technology up one language at a time (based on their order of implementation) enables you to check that each level works perfectly fine with all other technologies disabled. So if scripting is available, it will work, but it won't impact the already-tested functioning HTML and CSS.

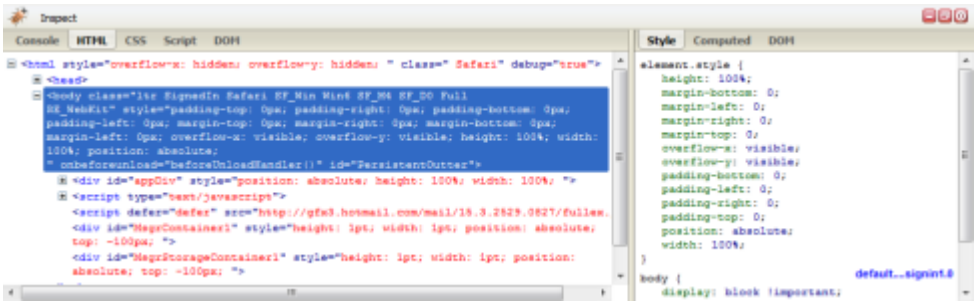

Figure 3.10. You can tweak your code on the fly using the Firebug tool

Bulletproof Layering

Bulletproof layering is ensuring that the layers that function below that language still work before an enhancement is added. And yes, I qualify everything—images, CSS, JavaScript, and Flash—as enhancements due to their inessential nature. It may sound harsh, but anything that can be disabled qualifies as optional.

The necessary elements of a page (according to browsers):

- page content

www.it-ebooks.info

- default element style
- attribute popups

The layering system only works if you spend the time to craft what you have at your disposal with the utmost care at every level. The value comes in a cross-compatible website.

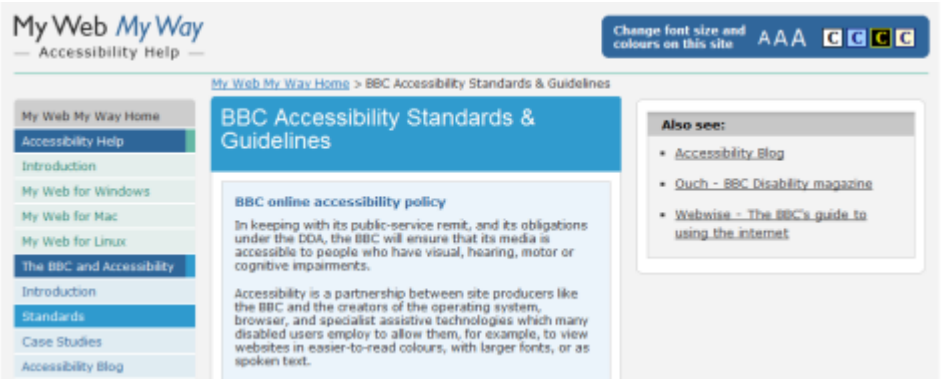
My Web My Way
Accessibility Help
Change font size and colours on this site
AAA
My Web My Way Home > BBC Accessibility Standards & Guidelines
My Web My Way Home
Accessibility Help
Introduction
My Web for Windows
My Web for Mac
My Web for Linux
The BBC and Accessibility
Introduction
Standards
Case Studies
Accessibility Blog
BBC Accessibility Standards & Guidelines
BBC online accessibility policy
In keeping with its public-service remit, and its obligations under the DDA, the BBC will ensure that its media is accessible to people who have visual, hearing, motor or cognitive impairments.
Accessibility is a partnership between site producers like the BBC and the creators of the operating system, browser, and specialist assistive technologies which many disabled users employ to allow them, for example, to view websites in easier-to-read colours, with larger fonts, or as spoken text.
Also see:
Accessibility Blog
Ouch - BBC Disability magazine
Webwise - The BBC's guide to using the internet

Figure 3.11. The BBC's specification on accessibility

It Matters to Your Users

It's important to note that while such testing may take more time to implement, it's worth the effort guaranteeing an accessible experience for your users.

The Risk of Dependence

The risk of dependence leaves many sites (even large ones) to this day with issues that plague their users. This can result in a loss of customers or site visits, with the back button possibly being the most clicked element on your site. The point is, gracefully degrading code is relatively simple to achieve, and almost maintains itself because of its fail-safe mechanisms.

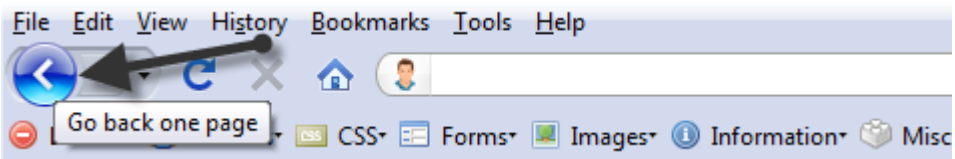

Figure 3.12. This may well be the most-used button in the World Wide Web

www.it-ebooks.info

The more visitors you gain and the smoother their experience, the more likely you are to maintain their attention. If you can do a job correctly from the beginning, saving yourself the problem of having to later patch against crippling bugs, you give yourself the best chance of success.

Figure 3.13. Mobile WAP browsers are a clear example of these limitations

The Future of the Fallen

The days of forcing technologies upon the user are long gone. Those still clinging to the poor techniques of old stand to lose customers. It's not the role of those in the industry to preach in the same way that we do about table-based design. It's your choice as to how your site is built, though many argue professionals should know better.

You should avoid:

- browser sniffing, and then demanding users change what they use
- technology sniffing, and then requesting users enable functionality
- building websites entirely around plugin technologies like Flash

As the Web becomes more complex, we can look to the future to see how it all may change. While exciting times lay ahead, it falls upon that famous line quoted in Spiderman to sum up the situation: "with great power comes great responsibility." So don't cut off your visitors' heads!

www.it-ebooks.info

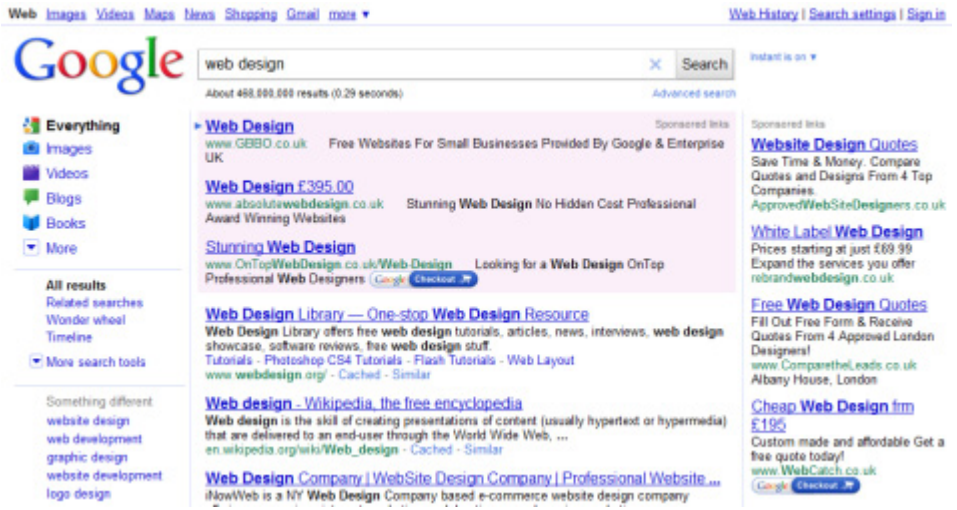
Web Images Videos Maps News Shopping Gmail more
Web History | Search settings | Sign in
Google
web design
Search
About 468,000,000 results (0.29 seconds)
Advanced search
Everything
Images
Videos
Blogs
Books
More
All results
Related searches
Wonder wheel
Timeline
More search tools
Something different
website design
web development
graphic design
website development
logo design
Web Design
www.GBBO.co.uk Free Websites For Small Businesses Provided By Google & Enterprise UK
Web Design £395.00
www.absolutewebdesign.co.uk Stunning Web Design No Hidden Cost Professional Award Winning Websites
Stunning Web Design
www.OnTopWebDesign.co.uk/Web-Design Looking for a Web Design OnTop Professional Web Designers
Web Design Library — One-stop Web Design Resource
Web Design Library offers free web design tutorials, articles, news, interviews, web design showcase, software reviews, free web design stuff.
Tutorials - Photoshop CS4 Tutorials - Flash Tutorials - Web Layout
www.webdesign.org/ - Cached - Similar
Web design - Wikipedia, the free encyclopedia
Web design is the skill of creating presentations of content (usually hypertext or hypermedia) that are delivered to an end-user through the World Wide Web, ...
en.wikipedia.org/wiki/Web_design - Cached - Similar
Web Design Company | WebSite Design Company | Professional Website ...
iNowWeb is a NY Web Design Company based e-commerce website design company
Website Design Quotes
Save Time & Money. Compare Quotes and Designs From 4 Top Companies
ApprovedWebSiteDesigners.co.uk
White Label Web Design
Prices starting at just £69.99 Expand the services you offer
rebrandwebdesign.co.uk
Free Web Design Quotes
Fill Out Free Form & Receive Quotes From 4 Approved London Designers!
www.ComparetheLeads.co.uk
Albany House, London
Cheap Web Design frm £195
Custom made and affordable Get a free quote today!
www.WebCatch.co.uk

Figure 3.14. If you refuse to serve them, Google will help them find your rivals

Everything Must Go!

No matter how complex it gets for your site, it's worth considering that everything must go. The ability to continue working with known glitches is expected of a computer; why should a website be any different?

If you have an existing website, try turning things off in various combinations to see what happens. If everything seems reasonable, give yourself a pat on the back—otherwise, try fixing it. If you are building a new site, layer each piece of technology independently in an unobtrusive manner. It will leave your site flexible to your users' needs, and that's good for business!

www.it-ebooks.info

Chapter 4

Going Freelance

by John Borda

So you've decided to go freelance. Good for you! But how do you handle all the nontechie bits that will help make you successful? You could be the best web designer on the planet (and you're reading this book, so you must be on the right track), but if no one knows, you'll make very little money doing it. And, lets face it, most people won't appreciate the gallons of coffee and late nights that went into designing that slick website or writing that killer app. Here's a small tip: there's no need for them to know!

People are interested in what you can do for them, not the logistics of what's involved. This goes for every part of the sales pitch, through the development, and out the other side into the aftercare. Your clients have headaches of their own—they don't need yours as well. But if you can take some of theirs away, they will love you forever!

Scenario One: Reception

So you're meeting the decision-maker at last. You'll probably spend a bit of time at reception, because:

www.it-ebooks.info

- you're early
- they'll keep you waiting

Rather than rehearse a pitch, listen. How many calls is the receptionist fielding? Are they all about the same topic? Maybe the answer to that question needs to be prominently placed on their new website. If you get the chance, you can even ask him—he may even appreciate a visitor taking an interest in his work. Ask even if the answer's obvious (for example, a school's term times and event dates), because it never hurts to have other workers in the company on your side. If the person has a good relationship with the decision-maker, he might even be asked what he thought of the individual who just left ...

Remember, everyone's important—not just the decision-maker. And regardless of whether you work with a keyboard, scalpel, or monkey wrench, you're in the business of taking people's headaches away.

Share a Different Perspective

We're all unique. As readers, we might all be in the same profession but, given the same specification, we'd generate very different websites. One person might see matters in a way another may not consider, and vice versa. This is actually a very good thing—especially if you can offer it to your clients in some way.

Scenario Two: Shared Interest

I'm Gibraltarian. There's obviously more to this than just coming from a particular place, as I'll illustrate. I'm also interested in green technology—if I wasn't coding, I'd probably be working in that field.

Recently, I was doing a little random surfing, when I came across a different technology that might have an application back home in Gibraltar—but not quite the one the makers had envisaged. I also noticed that the website could use a more professional touch.

Rather than just wade in with a sales pitch, I forwarded their details to a contact who might be able to use them. Then I told the company what I'd just done, including my suggestion about the different application of their technology. Only then did I make a brief sales pitch.

www.it-ebooks.info

The whole process barely took much longer than a basic pitch, but I received a reply within a few hours thanking me for my interest. They explained that they weren't ready to upgrade their website at present, but were aware it needed some attention, and would consider me for the work in future.

So, what have I just done?

1. identified that their product can fill two needs, rather than just one
2. demonstrated an interest in their product, not just their money
3. provided them with a significant sales lead

In short, I've provided immense value for free. Somehow, I think that will make me a lot more memorable come the time they decide to revamp their website.

So, I've just presented two scenarios. In the first, it's not about you, and in the second, it very much is. However, in both cases, the client has prospered. And that's really the point: make sure your clients receive a better product, and benefits will come your way.

Make Your Enemy Your Friend

You probably know several web developers in your area. You might think that they're competing for the same work that you are—or are they? Take a closer look at them; try and meet them at networking events, or socially, if you can. Website design is varied and wide-ranging, and they may seek completely different clients to you. They may also specialize in other areas. Of the web devs in my area, one has passed on work to me because he was too busy, and I've gone into partnership with another so that we can use our skills to complement each other. I've also sent clients to someone who specialized in an area I don't offer at present. All because I went out of my way to find them and introduce myself. Besides, we can all learn from each other—that's what this book is about! Even just by looking through other portfolio websites you can pick up a few extra tricks yourself; I know I have.

I've even expanded on this to include other IT specialists in the area; they're often asked about website design, too, as people tend to lump all of IT together. I offer referral payments if I make a sale off a lead they've sent me, and yes, I've had to

www.it-ebooks.info

pay up several times! Conversely, I'm also asked about techie stuff unrelated to the Web, which I prefer to pass on.

So, get to know your competition really well—you might even like them!

A Second Pair of Eyes

A second pair of eyes is the most powerful editing tool in the world, so don't underestimate it. If you're a solo freelancer, this could be your spouse or partner—my wife advises me on readability and elements for female-targeted websites—or even your children if the site is aimed at that demographic. I've even received good advice from a 13-year-old on the topic of PHP in the SitePoint forums, so don't overlook the wealth of talent at your fingertips in cyberspace, either.

I learned about this particular editing tool back at university, when another student was having trouble with a program. It was obvious to me that one tiny line was missing, but he couldn't see it. As I said, because we're all unique, we see things differently to one another.

Keep It Simple

… And then simplify again! Life can be complicated enough, and "simple" is easy to explain to your clients. Some of my best clients are technophobes; they don't want a website, but they need it! With these clients, I avoid overdesigning the interface they use to update their website, and hide any unnecessary optional extras. And, yes, they do need to update the website themselves, after all, you don't want to be a copy editor for all your clients! Keep it simple, and they'll even recommend you to their technophobic friends! It's worth it for the expressions on their faces when they realize that the Web isn't that scary after all!

And speaking of scary …

Work for Nothing!

Yes, I really did say that! When I was starting out, a friend sent me an article about a nonprofit organization that needed a website to promote itself. It had no money, but publicity was on offer, so I did the site for them. The lady I dealt with is still one of my biggest fans, and was quite shameless about promoting me in further press articles; obviously, my link was on its website. This opportunity started my

www.it-ebooks.info

career off in the right direction. Since then, I've done other jobs for free, especially for causes I feel strongly about. You can't pay for the kind of publicity that's associated with helping charities—it can really help to launch a career. But pick your charities carefully, and make sure they're ones you'd contribute to anyway; that way, you'll always be motivated to maintain their websites. Make sure, also, that you always leave time for paid work.

That's just a selection of what I've picked up along the way. I hope they help you in your journey to make the most out of the Web!

www.it-ebooks.info

www.it-ebooks.info

Chapter 5

Successful PSD to HTML Freelancing

by Paul O'Brien

Converting a PSD (Photoshop image file) into working CSS/HTML is quite manageable, but there are some important considerations to be mindful of before, during, and after coding. My aim is to point out the pitfalls, and discuss what you can do to avoid some of them.

I'll also explore some coding techniques that will help with the actual task of conversion, but this won't be a full "slice and dice" tutorial. It's assumed that you have a basic knowledge of Photoshop (or similar paint package) and average CSS (Cascading Style Sheets) HTML skills.

In most cases, I'll be talking about one-off PSD-to-HTML template conversions rather than designing a whole site on an ongoing basis. This is typically the work that freelancers are offered, and indeed there are many PSD-to-HTML conversion sites around the Web offering this particular service.

www.it-ebooks.info

Before You Start Coding

Before you start coding, there are a number of points you should raise with your client. It's useful to have a small checklist that highlights the main points for consideration, such as:

Rule 1 Always see the work first before quoting.

Rule 2 Establish what kind of work is required. For example, is it CSS and HTML only, or is scripting such as JavaScript or PHP required?

Rule 3 Discover what browser support is needed. This question is vital; otherwise you could end up having to code for IE4 because you failed to account for it!

Rule 4 Verify the timescales, such as the level of urgency, and how long the work will take to complete.

Rule 5 Confirm how much the job will cost.

Rule 1: Always See the Work First Before Quoting

I've been a freelancer for quite a few years, and usually the first approach a client makes goes like this:

> How much do you charge to convert a PSD to HTML?

This is much like the perennial question, "How long is a piece of string?"

My stock answer is, "It depends on what the PSD looks like."

Before you quote a job, you must look at the work involved in detail, otherwise you could be working for months for next to nothing. The variety of PSDs that I'm sent is astounding, and no two are ever the same. I did a small PSD for a client that was completed in about a day, and a few weeks later was asked to code another one for a similar price. On looking at the second PSD, it was immediately obvious that this one was much more involved; the design would take a couple of weeks to complete as the home page was about ten normal pages long with multiple stylized sections!

Although it may sound obvious, you must make sure before you start that both you and the client understand what's involved, and exactly what work you're undertak-

www.it-ebooks.info

ing. It's too late once work has started because the parameters of your employment must be set from the outset.

Rule 2: Establish What Kind of Work Is Required

Now that you've seen the PSD, you need to define what kind of work you are undertaking.

In most cases, a client will simply want you to convert the PSD into a working CSS/HTML template that they can then develop fully into a working website. You must, however, make sure that the scope of your work is understood from the beginning because scope creep[1] can become a nightmare to control.

Never assume that the client knows what they're doing!

The best projects are the ones where both you and your client understand exactly where the boundaries are and the extent of the work being undertaken. The worst projects are where the client expects everything he's ever seen on the Web to be included in the template at no extra cost.

To be honest, most clients fall somewhere in the middle, and while they won't intentionally expand the scope of the project, you will often be asked to perform a task outside the original scope. For example, "Can you just add this drop-down menu?" or "That looks good but can you revise the page and make it 100 pixels wider, and scale all the other elements as well"?

If you have already set the scope, it's easier to say, "Yes, of course, but it will take me longer and add to the cost."

I always stipulate that I'm just doing front-end work with no scripting or server-side work involved. That, of course, doesn't mean you should do the same; if your strengths are JavaScript or PHP, make it known at the start, and document the amount of work you're prepared to undertake. No one likes nasty surprises when working, so it's best for both you and your client to understand what's covered before you commence.

Remember to actually look at the structure of the PSD and the proposed structure of the HTML. If you have concerns that what the graphic designer has envisioned

[1] http://en.wikipedia.org/wiki/Scope_creep

www.it-ebooks.info

is a complete waste of time on a web page, you must discuss those issues and what needs to be changed at the time. Many graphic designers only have a small understanding of web pages, and it's your duty, as the coder, to look at the design and say whether or not it will work.

Once you've seen the PSD, ask the client how they want to present the source order of the document; for example, how would they like to tab through the page, and which content is the most important. Obviously, the end result does depend on the design in question, but you should at least know what's expected from the start.

Rule 3: Discover What Browser Support Is Needed

Although somewhat covered in the previous rule, I want to stress this point because experience has taught me that *I must stipulate to the client which browsers I am supporting*.

This is the number one problem freelancers have with their clients. Time and time again, I see the same issues appearing in the forum, where a client has switched on his old computer and complained that the coded pages aren't displaying properly in IE5.

At this stage, it's already too late to address the problem—the only way to prevent this scenario is to discuss it before you start. A number of clients may not even know what a browser is, but it's your duty to educate and inform them so that when they switch on an old computer, they'll realize that it's not a major problem if the site is slightly askew in IE5.

Forewarned is forearmed, and discussing with your client which browsers you are going to support is a worthwhile exercise that will save a lot of headaches later.

If the client insists that certain outdated browsers are to be supported and you're unable to talk them out of it, price the job accordingly, taking into account the extra time needed. When it comes down to cost, it's funny how quickly people can change their minds.

The question of which browsers you should actually support is, of course, dependent on the site at hand. For a new site, make an educated guess at the expected user base and select your browsers accordingly. If it's an existing site, look at the statistics

www.it-ebooks.info

in the server logs of the site and see which browsers have been visiting. If 99% of visitors were using IE6, you'd be wise to support IE6 among your list of browsers.

At the minimum, you need to make your page work in this current crop of modern browsers:

- Safari 3+
- Chrome 3+
- Firefox 3+
- Internet Explorer 6+
- Opera 10+

Which actual version you support is up to you and your client to decide, but as long as you have it in writing from the start, you'll both be covered. That way, if the client notices that the site fails to work in Safari 4, for example, you'll know that it's up to you to offer a fix.

Keep in mind that some of these browsers behave differently whether on a PC or Mac system, so you may need to devise systems for checking these platforms. We'll talk more about browser testing later on and what the current options are.

Remember that browser testing is an important phase of your job and not to be undertaken lightly. That's why I always test in various browsers while developing the page. At each stage or on the completion of specific elements or sections, you should test in the browsers that you want to support—just in case you build your page on a feature (or bug) of a specific system that's unable to be reproduced in other browsers.

In fact, I go a little overboard with this and will test every couple of lines of code just to be sure. I have about half a dozen browsers open at the same time, all pointing to the page in production. Once I've coded a few lines, I quickly click refresh in each browser to see what the page looks like. This takes seconds to do and is the easiest way to spot bugs and differences at an early stage.

Catching bugs early (the early bird catches the worm) means that in most cases a simple change in design at this stage can accommodate various browsers without using hacks. Waiting until you've completed a page and then doing your browser testing is a recipe for disaster. I'm surprised today to still see competent coders making this mistake.

www.it-ebooks.info

Rule 4: Verify the Timescales

Nine times out of a ten, a client will want the page completed by yesterday.

It's sad, but a fact of life. Clients spend months meeting with a graphic designer, discussing the project and finally achieving the design that they want, only to have made no provision for the page to be coded. It would be nice to be consulted earlier on in the process or to have some input into the design, so that it can be coded more efficiently—unfortunately, it's happens rarely.

In most cases, we freelancers are presented with a *fait accompli* and have to make the best of what's been handed to us.

The client is only interested in knowing when their page is going to be completed and will push you to hurry up, but you must specify a timescale that will realistically allow the work to be completed. Otherwise, you face the prospect of a client ringing you up every hour of the day asking, "Are we there yet" like some child grizzling in the back seat of a car.

If you *can* do it by tomorrow, it's fine to let the client know. Once you've made the decision, though, you must deliver as promised—so be realistic with your timescales. Saying you'll do something and then failing to deliver is commercial suicide, so avoid such a situation.

Just be clear up front about when the page is to be delivered and stick to it.

How long it takes you to code a PSD will depend upon your coding experience and the size of the PSD. As I've mentioned, some PSDs can be completed in one day while others can take two weeks or more; it's simply down to size or the number of differently styled elements. You'll need to take all these factors into consideration when determining the price to charge.

Confirm How Much the Job Will Cost

Finally, we reach the client's first question. How much?

If you've been through each step already, you now have a good idea of what's needed and how long it will take to complete. I always base my quote on how long I think it is going to take me and then price accordingly.

www.it-ebooks.info

To decide how much you can charge depends on a number of factors and is beyond the scope of this article. Look around and see what others are charging for similar work, as there are a number of sites that "look to hire" people such as the marketplace section at SitePoint.[2]

You should also look at the current hourly rate of similar industries in your area. Most people know what the minimum hourly wage is, and as a freelancer you should be quoting at least that.

Aim to charge a fair price at the going rate. Even if you are doing this for a hobby, you should not undercut existing designers—this just creates a downward (and unrealistic) spiral of expectation.

Obviously, when starting out you price yourself at the lower end of the spectrum, but as you gain more experience your prices should reflect your knowledge accordingly. Try to build a base of trusted clients who know that they can rely on you and what to expect. Life becomes easier this way, as clients tend to stick with you.

On the other hand, avoid clients that are too demanding time- and work-wise unless they're prepared to pay a premium. There's no point working for nothing, and while none of us like turning work down, you will occasionally have to draw the line.

Rather than say no to a client, I tend to raise my prices instead and let money do the talking. If they still want to go ahead, at least I'm being paid well for it.

As you can see, there are a lot of aspects to consider before you start coding, but I've tried to cover the main issues and now you can actually start to code.

The final part of the process it to put all the above in writing, then ask your client to sign and agree to it before work can begin. At the very least, ask the client to confirm by email that the details are correct if there is urgency to the project.

For new clients, you may also require an up-front payment if you know little about the client or if they have not been recommended from a trusted source. Never be afraid to ask for payment—after all, that's why you're doing this.

[2] http://sitepointmarket.com/

www.it-ebooks.info

Tools of the Trade

As coders, we all have our tools of the trade to make life easier when coding. From simple text editors (for example, Notepad/Notepad++) to more advanced ones such as Coda[3] for Mac systems, Style Master[4], or even Dreamweaver[5], which has a good code editor.

Whatever you use is fine, but you really do need to use a tool that speeds up your workflow. If you're still coding in Notepad, I'd suggest trying out more advanced editors in order to produce work more quickly and cut down on the simple tasks. I prefer editors that highlight and format the code, as well as spell-check properties or unmatched HTML. This can help you work faster and stop you making silly typos along the way.

It goes without saying that you'll need a copy of Photoshop, as most designs are in Photoshop PSD format. I also use Fireworks[6], which in most cases will open Photoshop files, but you have to be careful as not all files are compatible or rendered correctly. I prefer working in Fireworks as it's easier to "slice and dice," but I'm going to assume that you have the necessary Photoshop skills anyway (given this is not a Photoshop tutorial anyway).

We already mentioned browser testing earlier, but if you've yet to do so, you should download as many browsers as you can get hold of for testing. Internet Explorer is rather difficult to test in various guises, but there are a number of utilities that will allow you to run different versions of IE on the same computer. These are easily found on Google or by asking in the SitePoint forums.[7]

There are also a number of services (paid and free) that will take screenshots of different browsers for you, which can be a useful addition to your browser-testing routines. It's tricky to debug a page from a screenshot, though, so there's no substitute for having the real browsers in front of you. That said, feel free to have a look at the following:

[3] http://www.panic.com/coda/
[4] http://www.westciv.com/style_master/
[5] http://www.adobe.com/products/dreamweaver/
[6] http://www.adobe.com/products/fireworks/
[7] http://www.sitepoint.com/forums/

www.it-ebooks.info

- Browsercam[8] (expensive, but thorough)
- Browsershots[9] (free)
- Spoon[10] (free)

There are many alternatives, so I encourage you to conduct your own research through Google to see what's available to make your life easier.

If you're serious about being a freelancer, you really need a PC *and* a Mac, as there's no real substitute for testing on the platform itself.

There's also the added problem of testing on the different Windows incarnations such as XP, Vista, and Windows 7. Luckily, as far as browsers go there is seldom a difference between them, but you may find that text is occasionally rendered differently and won't fit in the same way. This is one of the reasons why I always recommend allowing elements to breathe (that is, give them more room than they need); in places where measurements are critical, you must control the overflow correctly.

Designers like to squeeze text into little boxes and in Photoshop you can make anything fit by just pressing it a little, but there's no such luxury in a browser. The best you can do is adjust the letter spacing, but the results will still vary between browsers. It is at times like this when it's helpful to test in a real browser on the system that you want to support.

The setup I use is as follows:

- PC running XP, installed with all the modern browsers (see Rule 3 above) and running IE6.
- Laptop running Vista and Internet Explorer 7 natively, plus all other modern browsers
- Mac OS 10.6.4, with all modern Mac browsers installed. Parallels[11] is also installed so that I can run Windows 7 with the latest Internet Explorer version.

[8] http://www.browsercam.com/Default2.aspx
[9] http://www.browsercam.com/Default2.aspx
[10] http://www.spoon.net/browsers/
[11] http://www.parallels.com/

www.it-ebooks.info

What setup you use is up to you, but if you're serious and want to do this professionally, you must have the tools to do the job properly.

Getting Started

As a starting point for all projects, you should set up some default templates that you can just grab and start writing your code within minutes. At the least this should involve a suitable valid HTML page with default structure and links to default CSS files. If you use reset files, have all the code in place and ready to go.

A reset file can be as simple as having a few predefined styles to get you going, or you may wish for a more complicated file. In truth, a reset file[12] is unnecessary, but it does pay to address some elements at the beginning so that you start on an even keel. A lot of advanced coders will eschew reset files as unneeded bloat, but you should decide for yourself.

It's not important whether you address the default state of elements right at the start or only when you actually use them. The latter method saves on redundancy, because you just define what you need at the time you use it. Using a reset usually means that you set everything to zero, and then when you use it you set it to something more suitable—so you end up addressing the element twice instead of once.

For the inexperienced, however, a reset may be a good idea. If you're handing your template over to a client, it may be wise to reset certain elements to stop the client from breaking your template later on.

Here's the template I use to start a project:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Title goes here</title>
<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon">
<!-- Meta -->
<meta name="description" content="#">
<meta name="keywords" content="#">
```

[12] http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/

www.it-ebooks.info

```
<!-- CSS -->
<link href="main.css" rel="stylesheet"
  media="screen, projection" type="text/css">
<!--[if lte IE 7]>
<link href="iecss.css" rel="stylesheet"
  media="screen,projection" type="text/css">
<![endif]-->
</head>
<body id="home">
<div id="outer">
  <h1 class="logo"><a href="#">Replacement Text<em></em></a></h1>
  <div id="header">
    <ul id="nav">
      <li><strong>Home</strong></li>
      <li><a href="#">Link 1</a></li>
      <li><a href="#">Link 2 </a></li>
      <li><a href="#">Link 3</a></li>
    </ul>
  </div>
  <div id="main">
    <div id="content">
      <h2>Main Content</h2>
    </div>
    <div id="sidebar">
      <h3>Side Column</h3>
    </div>
  </div>
  <div id="footer">
    <p class="copy">Copyright JG Building Services Ltd 2010</p>
  </div>
</div>
</body>
</html>
```

And here's some basic CSS to get started:

```
html, body, table{
  margin: 0;
  padding: 0;
  font-size: 100%;
}
object, embed, iframe,
h1, h2, h3, h4, h5, h6, p,
dl, dt, dd, ol, ul, fieldset,
```

www.it-ebooks.info

```
form, legend, table, caption, th, td {
  margin: 0;
  padding: 0;
}
ol, ul {list-style: none;}
/* end reset styles */
/* clearing technique */
.clearfix:after {
  content:".";
  display:block;
  height:0;
  clear:both;
  visibility:hidden;
}
.clearfix {display:inline-block;}
/* mac hide \*/
.clearfix {display: block;}
 /* End hide */
a img,img{border:none}/* remove borders from linked images*/
a:link{}
a:visited{}
a:hover{}
a:focus{}
a:active{}
hr{display:none}/* use for accessibility when css is switched off*/
input, textarea,select{
  /*font-family:Arial; set font-family for IE*/
  vertical-align:middle;
}
```

You should make your own templates based on the way that you work and add the elements you find yourself using all the time. There's no need to reinvent the wheel each time, and having everything to hand means that you can make a start straight away and get busy coding.

Getting Ready to Style

The first stage of conversion should be to grab your default HTML template and add as much of the content needed in order to give you a structure to work with. Enter key elements and headings in a logical fashion so that the HTML makes sense. For example, make sure all the headings are in place and make sense in the context that you have placed them, adding just a little content so that once you start styling

www.it-ebooks.info

you have something to work with. Avoid adding too much content at this stage in case the HTML needs to be changed later; although CSS is quite flexible, it does have limitations, and the markup may need to be in a specific order for it to work.

Structure Your Page Well

Some developers like to code all the HTML first so that they know they have a solid structure in place, because when all is said and done, the content is the most important element. Whether you go that far or not is up to you; either way, you should always let the HTML describe the structure of your content in the best way possible.

Most of the following is common sense, but here are a few pointers:

Do

- use heading elements (`h1`, `h2`, `h3`, and so on) for headings, but use them in a logical order
- use `p` elements for paragraphs
- use `ul` for lists such as menus and navigation
- use `ol` for when you have lists that must be in a specific order
- use the element that best describes the content you're presenting
- validate the HTML and CSS regularly to check for errors and typos
- remember that validation is a means to an end and not just a badge to wear

Don't

- use the break element (`br`) just to make space
- use bold to make heading text (such as `<p><b>Heading</b></p>`)
- add `divs` unless they're necessary or adding structural semantics to the page
- mix inline and block elements at the same level (for example, `<div><strong>Some</strong><p>Text</p></div>`). While it's still valid, it is not best practice, and indeed causes issues in older browsers such as IE6.

www.it-ebooks.info

- make all the images transparent. IE6 can't handle full transparency so, where possible, limit the number of transparent PNGs that you use. There are some solutions to the IE6 problem, but all have their downsides.

Slicing and Dicing

Rather than go into depth on the mechanics of slicing and dicing, I'm going to make a few simple observations that will save you time. Most designs will fit into a certain pattern, and it's your job to work out what the best pattern is for the design you're confronted with.

We'll take a quick look at a very simple PSD, the kind a small business would be looking to process, shown in Figure 5.1.

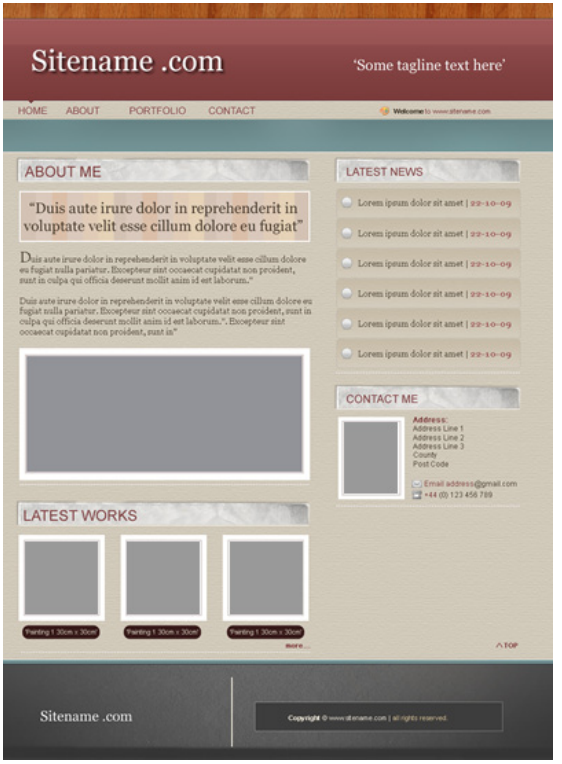

Figure 5.1. An example PSD file

This can easily be broken down as follows:

www.it-ebooks.info

- header section (also containing the main navigation)
- two floating columns (the main content and the sidebar)
- footer

Don't worry if there are some overlaps, as you can always move individual elements around, but avoid having multiple disorganized elements that are almost separate islands of content. Try to create logical groupings—rather than floating one bit right, and then another bit left, and then another bit right, and so on. You'll be surprised that even complicated designs can fit into quite simple structures.

Sometimes it helps to draw a wireframe over the PSD, as shown in Figure 5.2.

Figure 5.2. Overlaying the design with a wireframe

www.it-ebooks.info

Once you've simplified the design into these smaller pieces, coding the layout doesn't seem so overwhelming because you can do one little chunk at a time.

Where to start?

The approach I take is to start from the outermost element and work inside; then start from top to bottom. For the preceding site I'd begin like this.

1. Slice the repeating background image for the body and apply it to the body element. Set appropriate text colors for the site to match and ensure contrast. If the background is a repeating slice, make sure it does so seamlessly.

   I like to get the body background up and running, as it's a quick process and makes it look as though I've at least made a good start.

2. Next, create the page wrapper that's going to hold all your content. Some developers like to set a width on the body to do this, but I usually avoid that for a number of historic reasons: IE6 and IE5 are very buggy with this method, and it also throws the zoom out in IE7. Therefore, I much prefer to use one extra div and create a page wrapper to hold all my content.

3. Once you have your page wrapper in place, start from the top and work down, creating everything in logical order. Start with the header and work down the page as required.

The header can be static (not floated or positioned), as can the menu, although you may need to float[13] the menu items themselves.

The two columns should be floated, and it's best practice in a two-column layout to float both columns rather than floating one and defining the other with a margin. IE6 suffers from the 3px jog[14] on static content next to a float, and there would also be an issue of clearing content in the static column without also clearing the first floated columns. So use two floats and avoid all those issues, but remember to set your footer to `clear:both` so that it starts under the columns and doesn't wrap.

My CSS stylesheet also mimics this process in that I start from the top and work down. I start with a few defaults, working from the body element to the wrapper

[13] See the SitePoint reference for more on this. [http://reference.sitepoint.com/css/float]
[14] http://www.positioniseverything.net/explorer/threepxtest.html

www.it-ebooks.info

element, and then work down the page accordingly. I like to structure the stylesheet so that I know headers styles will be at the top of the CSS file and footer styles will be near the end.

Feel free to use your own methodology, but make sure it's logical and consistent.

Graphic Design Considerations

As you progress through the design, you may need to make compromises and changes; the main thing to consider is the content that's been provided to you. Graphic designers love for everything to be equal, and for it to just fit; hence, they'll quite often "massage" the content so that it lines up with another element exactly.

More often than not, the sections they're lining up are never going to hold the limited amount of text shown in the PSD. Therefore, at every stage you have to ask yourself questions such as:

- What happens if there is more content here?
- What happens if the text in the header wraps to another line because the browser's text size has been increased or some headings are longer?
- What if the user is using larger fonts or has changed the DPI settings?

Don't assume that what you see on the PSD will be true in a real-life situation. Content always changes and you must make allowances for this as you go.

Let Me Breathe

Avoid cramming text into tight spaces with pixel precision. Allow some breathing space to cater for browser differences. Don't set the height on containers that hold text content, otherwise they will break on text resize. If you must have a set height to match another element, use ems,[15] so that the element can grow when the text is resized.

If you allow some space in your designs, that will allow content to be slightly bigger or smaller without causing a problem.

[15] http://reference.sitepoint.com/css/lengthunits/

www.it-ebooks.info

Check That Your Content Fits Its Container

When slicing elements, always try to find ways to repeat graphics where possible, rather than slicing one massive graphic to form a background. Use `repeat-x` and `repeat-y` to create smaller images, but don't overdo it! Sometimes, it's easier to make a small image in one attempt, rather than ten complicated slices, so you need to think about what's best for the task in hand. There is no specific right way; it depends exactly on what's happening in the page and what you want to do next.

Create Appropriate Slices

When creating repeating graphics, don't assume that the smallest sized image will be the quickest method; for instance, a 1×1px image will have to be repeated many thousands of times by the browser in order to paint the whole screen. It's much smarter to make the image a little larger and have the browsers do less work. An image that is 5×50px will paint the screen 250 times quicker than a 1×1px image, with very little change to the file size in most cases. Experiment to find the optimal size for the task in hand.

It's worth pointing out a common mistake at this stage, and that's creating fixed-width graphics for text content. If you look at Figure 5.3, the caption within it is placed in a nice rounded-corner box.

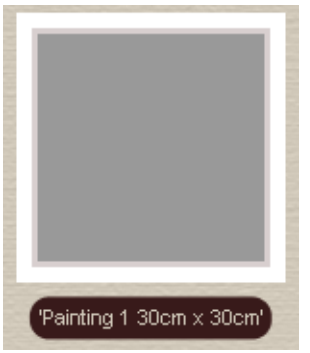

Figure 5.3. An image with a brief caption within it

You may be tempted to immediately slice this as one fixed-width-and-height image, like the one in Figure 5.4.

www.it-ebooks.info

Figure 5.4. A simple background image for the caption

Now the graphic designer has inserted only placeholder text, so what happens when the real image names are entered into that little box on the actual site?

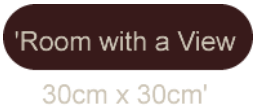

Figure 5.5. The caption overflows the box if more text is added

Once the text is entered, it exceeds the container, where it is barely visible on the background.

A similar effect would occur even with captions that fit if a user were to resize the text from the browser controls and zoom the text only.

It is your job to decide how to handle the overflow, and whether you need to refer to your client to discuss the problem (which should have been spotted at first sight of the PSD).

The options available to you are:

1. Do nothing (it's not your fault—you didn't design it).
2. Create a sliding door[16] button that will grow with text content.
3. Ask for a revised design to take into account text resize and text content.

By choosing the first option, you leave yourself open to having a badly designed website in your portfolio and an unhappy client—neither of which are pleasant.

Choosing the third option may be a decent compromise; you could ask the client if a simple square box could be used instead, which could be done without images and easily enable the box to grow as required. You could add some rounded corners using CSS3 `border-radius`[17] for modern browsers such as IE9, Firefox, Safari, to

[16] http://www.alistapart.com/articles/slidingdoors/
[17] http://reference.sitepoint.com/css/moz-border-radius

www.it-ebooks.info

name a few. IE8 and under have no support for `border-radius` (or a vendor-specific[18] equivalent), though, and would just get square corners but without loss of use or function. This is a reasonable compromise that would improve with time as more browsers supported `border-radius`.

For the time being, though, the second option is likely to be the best: it enables you to create an element that caters for varying amounts of text and allows for the text to be resized by the visitor. It's a lot more work, but the results are much better and everyone is catered for.

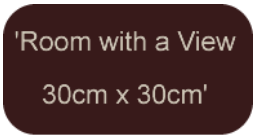

Figure 5.6. A flexible box can accommodate any amount of text

The element can now grow and shrink as required. As well as the sliding door approach, it can be constructed by one of the many rounded-corner techniques available.[19]

Use Sprites Where Possible

Most of the delay experienced in viewing a website is the time taken to go back and forth between the browser and the server. If you can cut out some of these "handshakes," the speed of your site will improve considerably. The best way to do this is to use sprites where at all possible, and combine multiple images into one background image that you can position via the `background-position` property.

Don't Account for Every Pixel

When you have, say, two columns of 450px width in a 960px-wide layout, don't account for the whole 960px in your calculations. Make both columns 450px wide, of course, but avoid making the gap between them part of your calculations.

[18] http://reference.sitepoint.com/css/vendorspecific
[19] http://tools.sitepoint.com/spanky/

www.it-ebooks.info

For example, you may think that you can float the first column left and then give it a 60px margin at the right to offset the second floated column (450 + 60 + 450 = 960).

That's fine on paper, but if lots of dynamic data is then entered into your columns, you may find that a 451px-wide image (for example) has been placed in one of the columns in error. Most browsers will cope with this and let the image spill out without affecting anything, but not IE6. Instead, it will increase the parent column to accommodate the larger image, resulting in the total width now exceeding the width available, and one column dropping down to find the space it needs.

As an alternative, we could just float one column to the left and the other column to the right, and not use any margins. This leaves us with a 60px gutter (free space) between the columns, which will act as a safety net for our old friend IE6. In the preceding scenario, the larger image would not cause the layout to break, although it would still cause one column to be wider.

What font is it?

Another trap that the graphic designer can set for us is the use of weird and wonderful fonts all over the shop. This creates quite a big problem, as we only have a limited amount of fonts to work with on the Web. Chances are that the fonts used aren't licensed for embedding[20], so we're not permitted to use them.

If you're unable to find a matching free font to embed, your choices are limited to:

- SIFR (Scalable Inman Flash Replacement)[21]
- Cufon[22]
- one of the standard font stacks available to all[23]

[20] http://reference.sitepoint.com/css/at-fontface
[21] http://www.mikeindustries.com/blog/sifr/
[22] http://github.com/sorccu/cufon/wiki
[23] http://articles.sitepoint.com/article/eight-definitive-font-stacks

www.it-ebooks.info

Font Embedding

For more on font embedding, Paul Irish's 2009 article on the topic[24] is a must-read. The process is simplified using the Font Squirrel generator.[25]

Nine times out of ten, I've convinced the designer to substitute a standard font stack instead. This avoids all the issues and overheads that the above methods produce and still creates a visually rich experience.

Avoid the temptation of using images for all your text; it would be a nightmare to maintain and very bad for all concerned. It's okay to use an image for the odd heading as long as you use a suitable image replacement technique[26], but it soon becomes too hard to manage when multiple images are concerned.

The easiest approach is to use normal browser text wherever possible for a more robust website. You can always fancy up the text for CSS3-compatible browsers using the `text-shadow` property,[27] which will allow the text to look good in capable browsers without a loss of function in others.

Optimize Images

One of the main reasons why a site can be slow is that the images used on it are simply too large. Images must be optimized to reduce their file size, and all good paint packages are able to do this. Most times, you can reduce the file size of an image considerably before you notice any real difference to the naked eye.

There is always a compromise between file size and quality, and a good coder will get it right while others will get it horribly wrong. There's nothing worse than visiting a site that takes ages for a pretty graphic to load when all you want is to access the content.

Avoid using very large images or slices where possible. It can be very awkward when the graphic designer has created a site with vertical, horizontal, and radial gradients, leaving the poor coder no choice but to use one massive image.

[24] http://paulirish.com/2009/bulletproof-font-face-implementation-syntax/
[25] http://www.fontsquirrel.com/fontface/generator/
[26] http://www.mezzoblue.com/tests/revised-image-replacement/
[27] http://reference.sitepoint.com/css/text-shadow

www.it-ebooks.info

These types of issues should be spotted when you first see the PSD. It should be obvious that the design won't work as a web page and should be returned to the graphic designer for alterations. In many cases, a slight change to the background gradient will enable it to be repeated, saving the use of that 200kB image or similar.

Final Thoughts

A lot of PSD conversion is about thinking on your feet, and asking yourself "what if?" throughout the process. It's about looking ahead and seeing the dangers, and then dealing with them before they're upon you. Of course, experience helps, but following the right path will help to ensure that obstacles are soon overcome and managed correctly.

Make sure that before you sign off on the page, you have a final check in as many browsers as you can manage, and validate the CSS and HTML so that there are no typos or errors. In addition, any content in the site should be run through a spell-checker; it looks unprofessional to regurgitate the spelling mistakes of the original designer.

By following these guidelines, you should successfully manage to convert a PSD into a functional web page that works across all browsers.

www.it-ebooks.info

www.it-ebooks.info

Chapter 6

Write Email Markup That Doesn't Explode in the Inbox

by Coyote Holmberg

Once upon a time, in the early years of the Internet (and even into the start of the 21st century), plain-text mailings held a place of prominence in email and email marketing. Most internet users had slow dial-up connections, images took forever to load, and email client support for HTML and images was sporadic at best. That was then, this is now. Today, the improvement and increase in availability of high-speed internet and the capabilities of email clients to deliver rich content has completely transformed the HTML electronic newsletter and it's place in the marketing and communications arsenal.

HTML Email: the Power and the Glory

The HTML electronic newsletter is a powerful medium for individuals, businesses, and organizations of all sizes and types for reaching out to customers. It can provide timely and visually compelling messaging based upon the recipients' demographics and preferences at a relatively low cost. While plain-text mailings still have a valid

www.it-ebooks.info

place in your toolbox (and should always be included as an alternative format), the improved bandwidth and email-client support means that now you can send newsletters in richly formatted text, images, animations, and (in select cases) video content one might find on a website.

Coding and Sending: Danger Awaits

Once you begin developing these newsletters, however, you'll find that coding for email is a very different animal to coding a website; it can be far more complex and frustrating than originally anticipated. At the time of this writing, support for HTML5 and CSS3 is nearly nonexistent, while support for CSS2 is problematic at best. Webmail providers and email clients all handle HTML and CSS differently, with varying levels of support for CSS elements; none provide full standards-based support for CSS, scripting languages, or dynamic content. Support by web-based email clients is also impacted by browser choice and version, so your newsletter's appearance and layout may display differently depending on which browser is being used.

Adding to the challenge is the need to evaluate both markup and content for spamminess, so as to minimize the risk of filters diverting your newsletter to the spam folder or blocking it completely. Last but not least, ensure that your business is following industry best practices as well as maintaining legal compliance with the CAN-SPAM Act. All of this means that sending your mailing from your own work email account is a bad idea, even for a small business.

Using the services of a professional email marketing provider is the easiest way of guaranteeing that your mailings meet the baseline requirements for both legal compliance and appearance. Features include customizable WYSYWIG templates, performance tracking and reporting, list management, and segmentation for targeted mailings. Many offer the options of using either their own templates, your own hand-coded HTML, or a combination to construct the HTML file, enabling greater control over the design and content delivery. Pricing options vary by the size and frequency of mailing, with some offering free accounts for very small subscriber lists (great for marketing your kid's summer lemonade stand).

www.it-ebooks.info

Email Design and Content: The Basics

Before you produce a single line of markup, you'll need to write and design your newsletter. Keeping certain core principles in mind during this process will increase the likelihood of a painless and successful campaign.

Subject Line

While not technically part of the code, a dynamic, compelling, and engaging subject line is important; without it, the most beautifully coded email in the world may not get read. The first thing a subscriber sees when your email shows in their inbox is the sender name and subject line. This is your introduction, the incentive you are providing for the subscriber to open and view your newsletter.

Some readers will simply delete emails if the subject line has no appeal for them. However, the attraction of the subject line must be balanced with the requirement to avoid triggers that can drastically increase your spam score. Spam filters work by assigning a rating to each element of an email newsletter (some items scoring higher than others), and the cumulative score determining whether or not your email is allowed entry or sent straight to the garbage can.

Call to Action

The call to action is the response to these questions: "Why are you sending this email? What do you want your reader to do?" Provide a clear message and a prominent place for the reader to click on your call to action.

Content Positioning

Prioritize your content, with the most important messaging of your email positioned above the fold—the top 400 pixels of the email for most email clients. This is prime real estate. Can the key point of your message be absorbed at a glance, even without scrolling? If your email is structured as a newsletter, with multiple articles or summaries, place a list of the articles (with links) in this area. If your email is predominantly image-based, this is where your most compelling or "hero" image should be positioned. Your primary call to action should also be located here.

The ideal email width is moderately flexible and steadily increasing. With all the different monitor settings, view pane configurations and sizes, and email client

www.it-ebooks.info

space allocations, it's tricky to gauge how your email will appear to your readers, though it may be possible to estimate this from your demographics. In order to maximize compatibility with the widest range of email clients (and reduce the possibility of the viewer having to scroll to read your email), your email design width should be ideally between 600 and 700 pixels wide.

A 2009 survey of large email marketers[1] showed that most were using a width of 600-650px for promotional emails, and 650-700px for more content-heavy (newsletter-type) emails.

In 2010, approximately 24% of computer users have a monitor resolution of 1024×768. Combining this with the view pane widths on the most commonly used email clients, if your email is 650px wide, just under 90% of your readers will be able to see the entire email width; the rest will have to scroll horizontally. At 700px, around 18% or so will have to scroll. If you keep your email under 625px, only 4-5% will have to scroll. These numbers will vary for you depending on your readership's demographics.

Email Isn't the Web

Throw out what you know about writing HTML and CSS for the Web. Coding an email requires a different approach to writing for a modern web page, even if the page and email appear identical on the surface. This is because of the varied and generally restricted support for scripting, dynamic content, HTML tags, and CSS formatting among email clients and web browsers. With a few exceptions (`text-align`, `font-weight`, and `color`), it's safe to say that for every styling and layout element, at least one email client won't support it.

If your goal is to create a mailing that provides a consistent experience across the widest range of email clients, it will be restricted to the most basic of elements: table-based layout, inline CSS, static images (or at most, animated GIFs), and the most commonly used (that is, preinstalled) web-safe fonts for your text. JavaScript, interactive content, Flash, video content, and other goodies are reserved for content on the Web, while spam filtering presents additional constraints.

[1] http://www.responsys.com/blogs/nsm/2009/08/email-width.html

www.it-ebooks.info

Coding Like It's 1999

When writing HTML for an email, it can feel like the early days of the Internet. Presentation and content are no longer separate. `<div>` tags are unreliable layout elements. Floats just don't. Z-indexing doesn't work. Header and external CSS vanish without warning. Fortunately, a few straightforward HTML code structures augmented with CSS techniques will deliver consistently stable HTML email designs, and make you look like a guru to your co-workers.

Step 1: Use Nested HTML Tables for Layout

CSS `float`, `margin`, and `padding` are poorly and inconsistently supported in Gmail, Outlook 2007, and Hotmail. The use of CSS for positioning of HTML elements is strongly supported in Apple Mail, AOL, and Mobile Me: all the other email clients have generally horrendous or (in the case of Outlook 2007) nonexistent support for CSS positioning. Because of this, tables are the only viable way to achieve stable layouts that don't explode. Begin with an outer frame `<table>` or `<div>` to serve as the HTML `<body>` tag substitute for the email (since some email clients ignore the `<body>` tag and/or strip CSS set in the `<body>` tag). Inside the `<table>` or `<div>`, create an inner `<table>` to function as the container into which you'll place all the elements of your email. At this point, your code will look a little like this:

```
<html>
<head>
  <title></title>
</head>
<body>
  <div style="margin:0px auto; width:100%;">
    <table cellpadding="0" cellspacing="0" width="625">
      <tr>
        <td> EMAIL CONTENT TO GO HERE </td>
      </tr>
    </table>
  </div>
</body>
</html>
```

Once you have your framework set up, you can begin determining the best way to slice the content for your email, based upon design elements and intended linking.

www.it-ebooks.info

Image mapping is not recommended for emails, so any portions of your design that are image-based and will be linked to urls need to be sliced into separate images; additionally, for ease and stability of layout, each image should be ideally placed into a separate table cell.

Set the `cell-padding` and `cell-spacing` in the `<table>` tag. Then set the width of each table cell in the `<td>` tag, and specify the width in pixels, not percents. Use table cells as needed to set left and right margins and padding, and nested tables to support complex designs, as width, margin, and padding within table cells and other HTML elements aren't consistently supported. `rowspan` and `colspan` in table cell layout and positioning are also inconsistently supported and may present unexpected design explosions: again, nested tables are your friend. Use `divs` sparingly for positioning, remembering that the `float` property is ignored by Outlook 2007. On the other hand, the `<span>` tag enjoys consistent and reliable support (being an inline element), and can be used to position text as well as resize or color type.

Beware Whitespace

Although you may use whitespace to make your HTML easier to read while coding, be sure to remove any extra whitespace between your `<td></td>` tags when complete. This is because Yahoo, Gmail, and Hotmail in particular may interpret that whitespace as additional padding and cause it to break (generally by adding a white line above or below the cell contents). This may be barely noticeable if your email is predominantly text and whitespace, or very bad if your email predominantly features sliced images.

Step 2: Writing Your CSS and Moving It Inline.

Next comes the fun part: writing the CSS for your mailing. This includes adding CSS fixes to resolve the various layouts predominantly used by email and webmail providers.

Gmail is the primary holdout for lack of support for the `<style>` element in either the `head` or `body` of the email, as well as for links to external stylesheets. This means that your CSS has to be inline, with the exception of CSS fixes targeted at specific email clients other than Gmail. Combined with the requirements to ensure Outlook support, your CSS should be applied at the `<td>` level or closer to the element being styled. This means that your font (and other) styling will need to be repeated in each table cell. Fortunately, there are free tools available that will move your

www.it-ebooks.info

header CSS inline, so it is okay to save time and redundancy by writing your CSS and classes as you normally would, and then moving it inline using one of these tools. I'd strongly recommend reviewing the generated code, though, for accuracy and for any tweaks made by email clients in how they render the code.

Step 3: Text Formatting and Paragraph Spacing

While CSS support for text and font formatting is fairly consistent, CSS3 and HTML5 support is virtually nonexistent (though Google promises that Gmail will support HTML5 in the future). To maintain readability and minimize any design "surprises," you're best off staying with with web-safe fonts; reserve any fancy use of typography for text elements embedded as part of an image. Text and paragraph spacing in an email is also subject to pitfalls, again due to the varying support for margin and padding, different treatment of line-height, and implicit spacing around paragraph tags. The solution: set line-height on all text, and reset your paragraph margins:

```
<p style="line-height: 12px, margin-top: 0px, margin-bottom:0px;
➥margin-left: 10px; margin-right:10px;">
```

Avoid using shorthand for your style and font notation, as this also lacks full support. For example, do not use the `font` shorthand declaration for setting properties:

```
<td style="color#333;
➥font:italic bold 12px Arial,Helvetica,sans-serif; line-height:0;">
```

Instead, break out and declare the properties individually:

```
<td style="color#333333; font-weight:bold; font-size:12px;
➥font-family: Arial,Helvetica,sans-serif; line-height:0;">
```

Set the background color for your email `body` and `table` elements, as some email clients default to colors other than white (or your preferred background color for your design). You wouldn't want your beautifully styled email in gray text to completely vanish in the default gray background of your recipient's email client. The solution:

```
<body bgcolor="ffffff">
<table style="background-color:#ffffff;">
```

www.it-ebooks.info

Links should also be formatted with inline styling, so as to preserve your preferred link color and underlining; otherwise, your links' appearance will default to the email client chosen color. Avoid the use of red fonts, as these can be a spam trigger. Remember to add `style="text-decoration:none;"` to the link tag and `border="0"` to linked images, to avoid surprise blue borders added to your email.

With emails that include both text and design elements, include space for line height and spacing variances, as this can display differently across clients and cause whitespace breaks.

Adding Images and Animation

A picture may be worth a thousand words, but it will certainly seem to require a thousand lines of code.

Besides using nested tables for layout, images present the next most challenging area for managing email client quirks. To begin, several webmail clients add unwanted spacing under images, resulting in white lines through the email. The code elements that will resolve this layout issue include adding `style="display:block;"` to each image and `style="line-height:0px;"` to each table cell. Make certain that you include height and width information for each image; otherwise, some email clients will use different sizes for your image when it's blocked, and your layout may be destroyed.

Unlike web pages, email doesn't support relative linking, so all images must be hosted and linked to with absolute (not relative) links. For example:

```
<img src="http://yoursite.com/images/email-image.jpg" height="200"
➥width="300" border="0" style="border:0px; display:block;"
➥alt="my email image">
```

Avoid Using Background Images

Use inline images for any primary content, as background image support is buggy, or only partially supported in Outlook 2007 and Gmail, and has no support in Hotmail. Any key images in the email should be placed inline and in your table cell (not as a background image). If your design does include a background image element, and you've acknowledged that it's okay for some users being unable to see it, include it twice: once on the `body` tag so that it will appear in Outlook 2007, and

www.it-ebooks.info

once on a wrapper `div` for Gmail. Include an appropriate background color to provide graceful degradation for Hotmail.

Prepare for Image Blocking and Missing Images

Some email clients block images by default, so it's vital that your content is readable even when the images aren't displayed. Each image should have the height and width set, as well as an `alt` tag containing a brief description of the image (or if the image contains text, the text contained in the image). If this is impractical due to the quantity of text in the image, at a minimum include a summary or the key points of the text. To further enhance the user experience, you can format this `alt` text for font size and color.

Go Ahead, Include Your Dancing Hamster Animations

… But only if they're in GIF format. Video and Flash support in email (or lack thereof) will make you feel like you're truly back in the dark ages of the Web. In a nutshell:

- While Apple Mail supports most media types, Flash, QuickTime videos, Windows Media Files, and Java Applets are not supported for the rest of the email universe. Don't even think about it.
- HTML5 Video embedding is currently working for Apple Mail and iPhone, with fallback content (such as a static JPG) visible for other email clients.
- Gmail will allow you to embed a link to YouTube videos in the email. At the bottom of the email, Gmail includes a thumbnail of the video and the "play" icon, making it appear as if it is an actual embedded video.
- Animated GIFs work in all email clients except Outlook 2007, which will only show the first frame of the GIF. This includes "video GIFs," in which the video for embedding in an email is a single large (200K+) animated GIF. Therefore, if you include animation, create it so that the key information or scene is contained in the first frame of the animation. If you're using large file size images, you may wish to determine how many of your site visitors using slower connections or equipment, as this could clog their inbox or affect download time for your email.

www.it-ebooks.info

Beware Dynamic Content

JavaScript, forms, search boxes, and other dynamic content types generally won't work in your email. Support for these types of content ranges from sporadic to nonexistent across email clients. Instead of embedding, link to your form or other dynamic content on a website for the best user experience. Besides being safer, more people will be able to use it and participate in your promotion.

Avoid using characters such as >, <, ", &, and so forth in `alt` tags or other text areas, as they may be read as HTML notation rather than the intended text element. If you must include these, use the appropriate character entity instead. For example, instead of >, use `>` for double quotes, use `"`, `”`, and `“` as required.

It's Coded. Now What?

Once you've finished building your email, the next step is to validate your markup (you don't want your layout to break because of a missing closing tag); then test the email for appearance, links, and broken or missing content in the various mail clients. You can do this either by creating a set of test accounts and sending a test copy of the email to each one, or by subscribing to a service that will test for you and provide you with an instant view of how the email appears. These services generally provide a series of screen captures, but not access to the generated source code from the email clients.

The advantage with these testing services is that they test against a wide range of clients and platforms, and can provide a complete set of results very quickly; these results would otherwise take a considerable amount of time if you were to use your own test accounts (you'd have to send the email, access test account on multiple browsers and platforms, review, rinse, repeat). The disadvantage is if your newsletter looks bad, you'll have to play at being Sherlock Holmes and determine what might have gone wrong, because they only provide screen captures—not the code or the web browser it broke in. The answer, sadly, is not always elementary.

Doing your own testing takes longer, but you get to view the source code as it appears in the recipient's mail client. Besides allowing you to test for elements such as broken links, this can be particularly useful for debugging markup from mailings that combine dynamically generated content and hand-coded HTML elements.

www.it-ebooks.info

Once testing is complete, you are ready to send. Congratulations!

Resources

- The Email Sender and Provider Coalition (ESPC)[2] was formed to fight spam while protecting the delivery of legitimate email.
- Net Applications[3] provides free global market share statistics on internet usage. It includes monthly information on key statistics such as browser trends, monitor resolution, and browsing by device (mobile versus desktop).
- The Litmus report[4] shows the current state of the email client market, and gives the market share for the top 10 email clients detected.
- The services provided by the nonprofit antispam Spamhaus Project[5] include maintaining a real-time block list of known spammers and a whitelist of verified legitimate email senders. It also provides an FAQ of recommended best practices for email marketers.
- The Email Standards Project[6] provides a report on the current state of web standards support in popular email clients. It is focused on working with email client developers and the Web and email design community to improve web standards support and accessibility in email. It aims to "create a better experience for everyone who creates, sends, and receives HTML emails from permission-based lists."
- Campaign Monitor[7] maintains a detailed list[8] of the current state of CSS support in email clients.

[2] http://www.espcoalition.org
[3] http://www.netmarketshare.com/
[4] http://litmusapp.com/resources/email-client-stats
[5] http://www.spamhaus.org/
[6] http://www.email-standards.org/
[7] http://campaignmonitor.com/
[8] http://www.campaignmonitor.com/css/

www.it-ebooks.info

www.it-ebooks.info

Chapter 7

Make Your Website Stand Out from the Crowd

by Ursula Comeau

Web design is further evolving now that online marketing has become such an important part of one's online presence. This development is the result of building relationships with others (whether they be friends or clients) and interacting one-on-one with them. Think back to the year 2000: we had just faced the Y2K bug, eBay, and Amazon were still in their infancy, having a website was a privilege of mainly businesses, and the term "blog" was almost unheard of. Online marketing was about having a web presence (in the form of a website), and perhaps sending out an occasional newsletter by email, in addition to printed media. Basically, one-way communication was the norm.

Fast-forward to 2010, and the word "spam" does not necessarily refer to a can of meat, and Facebook and Twitter have literally become household names around the world (especially households with teenagers). What has made Facebook and Twitter popular so fast? Interaction is the key—being able to build relationships with others, have conversations, and connect with people. Two-way communication

www.it-ebooks.info

is fast becoming the norm on the Internet, and anyone wanting to market themselves or their businesses needs to implement a social media strategy.

If you're a seasoned veteran of blogging and social media, read on—you just might gain some useful ideas! If you're new to this arena, you'll learn just how important social media and blogging can be to your website and your online marketing strategy.

As the world becomes more aware of social media outlets, requests for websites and web design often include a requirement for built-in blogs that include links to Twitter, LinkedIn, and Facebook, among other social media hubs. Web designers need to be prepared to provide these services and understand how they work, otherwise they risk losing business. A web designer doesn't need to necessarily understand how every social media option functions or know how to set it up for a client, but being aware of the major players, and being able to teach a client how to utilize social media with their website is vital in order to stay ahead of the game and maintain business. For those who build their own websites, this is important to know for their marketing strategy.

So what exactly is social media?

Social media is not as complicated as it may seem. The key word is *interaction*, and how it applies to one's online marketing strategy; this is the easiest way to determine whether something is social media. Using Merriam-Webster's online dictionary,[1] let's define the words individually:

Social Of or relating to human society, the interaction of the individual and the group, or the welfare of human beings as members of society

Media A medium of cultivation, conveyance, or expression

In other words, social relates to people interacting with each other, and media is the communication channel. Now, let's define interaction:

Interaction

Mutual or reciprocal action or influence

So putting it all together: social media is when people influence one another through communication in a given environment. Let's see if this definition holds up with a

[1] http://www.merriam-webster.com/dictionary/

www.it-ebooks.info

few of the major players. I'm going to try to describe (or even define) Twitter, Facebook, LinkedIn, and YouTube in my own words:

Twitter
: a network of people having micro conversations (up to 140 characters at a time) with each other via typed messages

Facebook
: a network of people conversing and sharing ideas, photos, videos, games, links, articles, and so on

LinkedIn
: a network of professionals having conversations, sharing expertise, and meeting other professionals

YouTube
: a network of people sharing videos and interacting with each other via videos, comments on videos, and/or video responses

All four of these social media sites involve people influencing one another through communication in a social environment.

Where does blogging fit into all of this?

Blogging is a form of social media because it involves people interacting via a website. Let's define the word blog:

Blog
: A website comprising an online personal journal with reflections, comments, and often hyperlinks provided by the writer.

The difference is that instead of interacting on someone else's website, the interaction happens on the blogger's own website via comments to articles the blogger has written. Additionally, blogging isn't necessarily one's personal journal anymore—many businesses now have blogs relating to their industries and areas of expertise to foster interaction with their customers, thereby driving more traffic to their websites.

www.it-ebooks.info

Driving Traffic

Social media will help play a role in building your credibility in your area of expertise. By using social media as part of your marketing strategy, you'll create a larger web presence than just your standard static website with information on different pages. Sure, implementing SEO (search engine optimization) will help drive traffic to your site from search engines, but saturating your content in the search engines takes time, not to mention the competition you face, along with new ones popping up every day.

With social media, you're more accessible to the public seeking answers, and can be seen to share resources relating to your niche; such factors serve to create trust among the community at large. It will play a part in building your brand and, to put it simply, marketing yourself to the public. It also helps to drive traffic to your site, because some of the content you share will be your own (perhaps from your blog), leading people to your website. Furthermore, if a user likes your content, it's plausible that they'll share it with others, enabling your content to spread with ease through word of mouth! Essentially, it's about building trust, and somehow making yourself memorable so that people return to your website and share your content. You'll need to pay heed to your writing style, too, as it will play a role in being memorable.

Creating a Memorable Experience: a Real-life Example

Imagine you're eating out at a restaurant and you're served by a waiter. If the waiter is average, you'll receive adequate service (and nothing to complain about), and you'll go home with a full stomach with any luck. Now, if your waiter is awesome and provides extraordinary service, you're likely to remember it, even choose to go back to that same restaurant in the future, and perhaps even request to be served by that waiter!

How would such a memorable experience be created? Such waiters know how to connect with guests, whether through humor, showing interest in their patrons' lives, being a little zany, or just by creating a comfortable environment. This type of waiter usually receives a bigger tip than the average waiter, as most patrons are happy to reward great service. Same as in the online world, by providing great service and content for your *patrons*—perhaps even being a little unique—you too can benefit from satisfied customers.

www.it-ebooks.info

So we've established how social media can drive traffic to your site, but having a blog can pay even bigger dividends. Fresh, dynamic content written in your unique voice will help strengthen your brand and make you memorable enough for people to keep returning. Remember that you are unique—there is no one else on the planet like you, so don't be afraid to express yourself. Blogging is not about writing content for a textbook; it's about connecting with people and grabbing their attention.

By the Book

How much attention did textbook material hold for you when you were in school? Unless you were a complete nerd who loved to learn about everything you could, probably not a lot. I certainly wasn't particularly overjoyed at the thought of reading several chapters of my science textbook, but I loved to read fiction novels during my spare time—even depriving myself of sleep just to finish a good story! That's the kind of dedication you want to aim for, but you won't achieve that without infusing your writing with your personality.

Keep this in mind when using social media and writing blog posts—it is meant to be an informal, easygoing communication medium, whether it's a personal or a business blog, or any social media strategy. Let the main website carry the more professional verbiage, while the blogs and social media blurbs adopt a more conversational tone, as if you were talking to a person in front of you. This will help you be more memorable, encouraging visitors to return.

Being Memorable

Making yourself memorable will require a conscious effort. We're all constantly bombarded by images and advertisements, and time is precious in our busy lives, so we're going to be very particular about how we spend it and what we read, especially online. This is part of the reason why Twitter is so popular. It takes very little time to read a few tweets comprising less than 140 characters, let alone post tweets. Using Twitter, in fact, is a form of blogging: microblogging. Microblogging can also direct people to full blog posts or other media you've posted on your website (such as audio podcasts or videos), so choosing a catchy title for a page or blog post is important. Here are some tips to keep in mind when creating content—whether it be a page, blog post, audio podcast, or video.

www.it-ebooks.info

Rules of Engagement in Blogging

Create a purposeful title

The aim here is for the title to grab people's attention, enough so that they're interested in reading your content. The above heading "Rules of Engagement in Blogging" is an example of a purposeful title using a play on words—we want to engage our readers after all! From here, your content needs to maintain their attention, but also keep SEO in mind when you're creating your title, so that traffic will also arrive via the search engines.

Let your voice shine through

Captivate your readers with your personality. You want them to stay reading your content once your awesome title has lured them in. Again, keep SEO in mind when writing your articles.

Satisfy a need through your content

The best way to hold a reader's interest is to provide some form of solution to a perceived problem, whether it's a question or an actual issue that exists.

Keep your content short and precise

It's important that your content isn't too long, especially to the point where people don't end up reading everything. If you do have a lot to say on a particular subject, turn it into a series (part 1, part 2, and so on). You may like to produce a list (like this one), as they're easy to both write and read.

Ask for interaction

Having a person read through your content in its entirety is an accomplishment in itself; remember, however, that social media is about interaction, so be sure to include questions within your blog posts that prompt readers to respond and leave comments.

Reply to comments

If you've done your job right and people are commenting on your written content, remember to respond to people's comments. It's important they know you're alive and listening, and that you care about their opinions as well.

Aim to be *respectfully* controversial

Posting controversial material can actually help to drive traffic to your blog or website, but be aware that you may receive less than desirable comments as a

www.it-ebooks.info

result. It's not a perfect world after all, and you'll want to be diligent in moderating your blog's comments.

You can also apply some of the above rules to other applications of social media, such as in your tweets, audio podcasts, videos, and even your comments/responses to another person's blog post or social media hub content.

Is having a blog absolutely necessary?

No, a blog isn't a necessity, but it will help drive more traffic to your website, especially from search engines. A blog has the dual purpose of providing a medium with which to interact with people online, as well as playing a role in search engine optimization. The more dynamic the content on your website is, the more indexed your site will be with the search engines, which means you have more of a chance of being found via search results. As long as you have good-quality content to share, it will help improve your visibility, and thus, your credibility.

Some of the benefits to having a blog are:

- having a low-cost marketing tool (the only cost is time!)
- gaining trust in your service/product offerings or area of expertise
- showing your accessibility via interaction on blog comments
- generating traffic to your website
- attracting search engine attention (part of SEO)
- using it to serve as your primary social media portal

By being able to *prove your expertise in your niche*, you help improve your credibility. If a reader feels confident in taking advice from you, they are more likely to buy your products or services, and continue to visit your site. Gaining trust is a key marketing factor.

Blogs can also act as feedback forums, enabling your readers to suggest what they'd like to see in your content, so that they keep them coming back and spreading the word about your site. Businesses especially can fail to fully realize how having a blog that promotes interaction makes a company more accessible and "real" (for lack of a better term) to others. Being approachable is essential when trying to really connect with people and build a more personal relationship.

www.it-ebooks.info

Consistently updating your blog puts you "on the map" with search engines; by using keywords relating to your niche, you increase your chances of ranking higher in searches, thereby driving more traffic to your website. This is a fundamental element of search engine optimization.

Indeed, a blog can form the central hub for your social media marketing strategy. It will act as your central connection to the social media websites you use, as well as being where your social media profiles link back to. Think of it as your social media home base.

The Role of SEO

Because time is precious, you may want to concentrate initially on mastering on-page search engine optimization. SEO is a broad subject, but for the purposes of a website or blog, on-page SEO is a good place to start.

On-page SEO is about using keywords in the content of your website's pages and blog posts. The importance of SEO lies in improving your website or blog's ranking on search engines, so that when people search for keywords you've used, your content shows up as a search result. So far, we've discussed the more personal aspect of gaining followers and traffic. SEO is just as important with the amount of searches done by people daily on search engines, and definitely shouldn't be ignored.

Here are some tips for implementing a social media strategy on your website or blog:

- Use keywords in your actual content that are relevant to your article or page.
- Repeat the keywords in your article or page content, especially those that you want to have best optimized (what people would type into a search engine to look for the content you're posting).
- Aim to have more than one related keyword in an article, as it will give you more chances to be found in a search result. Think of it this way: one sentence can be said in several ways, with different words, so one person may prefer one word over another. Be aware, though, that search engines are smart enough to not return too many results from your website or blog, so avoid exaggerating and having too many articles, posts, or pages that are optimized for the same keywords.

www.it-ebooks.info

- Don't stress out too much about including keywords. Instead, write your article as you normally would using your own words, but keep appropriate keywords in mind as you're writing. The point here is to not think of keywords first and then write your content; you want your content to sound natural and within context, not as if it was spewed out of a computer.

- The title of your content should contain important keywords. Solutions exist where you can set the title that shows up in search engine results to one that's different to the actual title of the post or page, which can be helpful for optimization.

- Using secondary titles within your article (such as HTML tags `<h2>` and `<h3>` for secondary titles or subtitles) can also help with your optimization, but again, make sure at least one keyword is in the subtitle. This will also serve to break up the content and make it easier to read; separators and extra whitespace in any form of documentation helps to keep readers' interest visually, and not overload them with too much text.

- Ensure that you use your keywords in the first paragraph of your article. Think of how an essay is written: the introductory paragraph tells the reader what the paper is about, and tries to stimulate interest, particularly by including a hook (thesis) in the last sentence to entice an audience to continue reading.

- Keywords can be single words, multiple words, or phrases. Think of how a user would type their search query into a search engine, and go from there.

- Make sure your blog is set up so that the title and the URL for the content also contains important keywords. For example, depending on the content management system you're using, the default may be for a page to appear as www.yourwebsite.com/ ?p=128 instead of www.yourwebsite.com/yourpage—you want to make sure you're using the second format.

- Link back to older content or posts on your site or blog (backlinks) that are related to your present article, again in a natural format.

- Using your keywords in the meta description is also important. The **META description** is a brief description of your page's content that's used as an advertisement in search results to indicate what your content is to searchers. And I do mean brief—Google, for example, only displays 160 characters, so think of the

www.it-ebooks.info

meta description as a mini-ad to entice people to click on your result and read your content. If you decide to not go with a meta description, search engines will extract the relevant text anyway when showing your content in search results; still, it's a good idea to have one if you want to try to guide the search results to your particular keywords.

Your end goal is to have your content show up as a first-page result (ideally) when a person searches for keywords relating to your website or blog. Implementing an on-page SEO strategy throughout your pages and/or posts will help you reach this goal.

Setting Up a Blog

By far, the most widely used and popular blogging solution is WordPress. It's user-friendly, flexible, widely supported, and regularly updated and maintained. By default, it's easy to optimize and create SEO-friendly content, but there are WordPress plugins to utilize should you wish to further increase your SEO rankings.

Originally, WordPress was designed to be a blogging platform, but constant improvements and developments have been made to expand its features and functions. There are two variations of WordPress available: the free WordPress.com version and the self-hosted WordPress.org version. In an interview, Matt Mullenweg, the founding developer,[2] used an analogy to describe the differences between the two options: WordPress.com is like renting an apartment, while WordPress.org, is like owning a house.

When you rent an apartment, all you really need to worry about (generally) is rent—everything else is taken care of for you (for example, if something breaks down, you call your landlord), but you have less say in any improvements or repairs. On the other hand, when you buy a house, you have the freedom to change its appearance, but everything is your responsibility: acquiring major appliances such as a stove, handling maintenance costs, and so on, as well as paying property taxes.

Likewise, if you just want a simple blog without the worry of maintaining the platform itself (setup, updates, and so on), you can use WordPress.com to set up your blog, even use your own domain (compared to something like yourdomain.wordpress.com). Maintenance of the WordPress platform itself is taken care of for you.

[2] http://wordpress.tv/2009/10/29/matt-mullenweg-wordpress-now/

www.it-ebooks.info

On the other hand, if you want to have more control, such as the additional functionality via freely available plugins, and the freedom to design the website as you like, you'd want to go with the WordPress.org option.

Web designers will usually go with the WordPress.org option, as they'd already have hosting set up for the website, and are comfortable with the technical side of installation, design, and maintenance. A self-hosted WordPress site needs to be kept up to date regularly, to maintain proper functionality, as well as adopting security fixes that are frequently made to the latest build (available from www.WordPress.org).

It is possible for a web developer to design the main site as a static website, and add on a blog using WordPress, or to design the full website (static content with dynamic content in the form of a blog or news updates) using WordPress for everything. Most often, if a client has an existing website that they're happy with, they'll opt to just add on a blog, although a complete re-design should be considered, depending on how dated their original site is.

It's a good idea for web developers and/or designers to become familiar with WordPress and how to implement a site based on the publishing platform. More and more people are looking to implement CMSes for their websites, and WordPress is by far the best solution to use. It does require some knowledge of PHP, MySQL, and CSS, but it is so easy to work with that it is possible to use WordPress without being an expert on any of these technologies.

A good place to learn how to work with WordPress is to visit the WordPress.org website, where you'll be able to download your own copy and install it on a web hosting server. There, you'll also find the instructions to the famous 5-minute installation[3] to get it going. A MySQL database will be required to run it, and the WordPress Codex[4] should be the next place you visit to learn more about how to maximize its potential.

[3] http://codex.wordpress.org/Installing_WordPress#Famous_5-Minute_Install
[4] http://codex.wordpress.org/

www.it-ebooks.info

In Summary

Research has shown that blogging and social media marketing are extremely important components to one's online presence, especially in the exceedingly competitive internet environment. Whatever area of business or niche you're in, competition is tough, and having a static website just isn't enough to help you stand out from the crowd anymore. Interacting with people through social media will help you gain the edge on other websites in your industry, while projecting your own voice through a regular blog will contribute to driving regular and new traffic to your site. You can even gauge your efforts by tracking your site's analytics, so that you can see the results of your efforts. Social media and blogging go hand-in-hand with having a website. They're not hard to work with, either; they just require your time and dedication.

www.it-ebooks.info

Chapter 8

Information Organization and the Web

by Sherry Curry

Let's say an organization or company has a great product or idea, but how well can its users access the information about it on the website? In this age of information overload, it's all too common for a website to be given only a cursory glance or to be overlooked completely. Information should be clearly organized and labeled to help identify and promote an organization's website. It should be presented in a structure that flows and is accessible to all. The principles and practices of information architecture are complex and form a vast field on their own. But the basics still apply, whether it's for large corporate websites or small, not-for-profit organizations. The following pages will give an overview of the information building process, as well as provide a few relevant ideas.

Getting Started

Before the actual design gets underway, the information structure should be developed. This involves becoming familiar with the organization's function and background. Every element must be considered: goals of the organization, purpose of the website, target audience, user needs, design elements, review of an existing site, comparison of competitive research, and more. If there is an existing website,

www.it-ebooks.info

determine what content needs to be carried over. What content needs to be edited or eliminated? Is it up to date? If not, what new content needs to be added?

Information elements should be ranked by priority and relevance within the structure. For example, an organization's hours should be located within the "About Us" category and not listed separately. The number of headings, categories, and pages should be determined. These should be kept to a minimum if possible (the "Rule of Six" suggests that no more than six bullets per page should be used). Users should not be overwhelmed by too many choices, and also be comfortable with the terminology used (which will be discussed later).

The process of selecting the information ingredients varies by developer, but involves such tools as card sorting, personas, mock workups of a web page (often called wireframes), and sitemap development. For more information, refer to Christina Wodke's SitePoint article "Boxes and Arrows: Defining Information Architecture Deliverables."[1] This is also discussed in Morville and Rosenfeld's highly regarded work, *Information Architecture for the World Wide Web* (O'Reilly Media, 3rd edition, 2006).

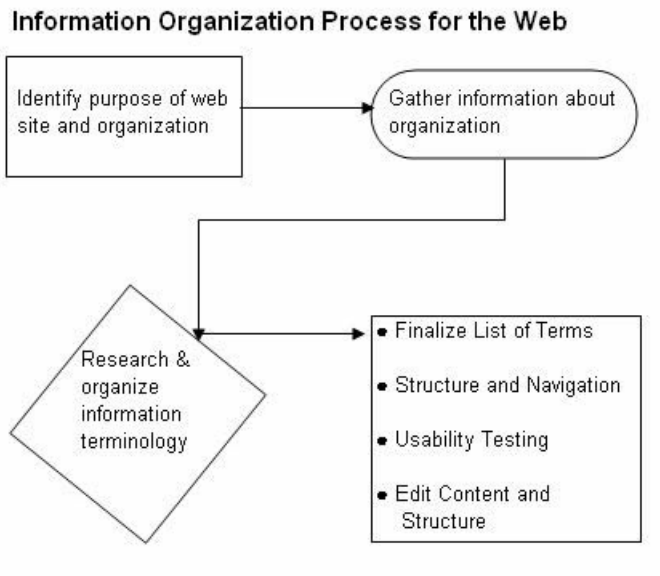
Information Organization Process for the Web
Identify purpose of web site and organization
Gather information about organization
Research & organize information terminology
Finalize List of Terms
Structure and Navigation
Usability Testing
Edit Content and Structure

Figure 8.1. Information process for the Web

[1] http://blogs.sitepoint.com/architecture-deliverables/

www.it-ebooks.info

Word Choice

Users should be able to understand and interpret the language of the website—how else would they be able to use the site? Indeed, how successfully would a search provider such as Google lead users to the site? The site's word choice, or **terminology**, should be both predictable, up-to-date, and consistent. For example, a maker of children's playground equipment should either consistently use the term "monkey bars" or "jungle gyms," but not both (while perhaps including a cross-reference for the unused term). They should also choose the most acceptable, up-to-date term (my kids prefer the term "monkey bars"). Finally, it's okay to use the nomenclature belonging to a specific audience, but term consistency is still important.

What does that term mean?

Proper organization of terms is key, but during the organization process, developers may be a little intimidated by a subject field they personally know little about. Suppose a developer is asked to design a site for an assistive technology company. To the developer, *JAWS* is the name of a great 1975 Steven Spielberg thriller. But to the company, JAWS is an assistive technology that helps the visually impaired reader interpret the words on a screen. Thus, the developer may need to further investigate the field, looking at company documents as well as other websites and resources, and conducting searches on the terminology. A developer doesn't have to know the subject; they just have to understand it.

Selected Resources on Terminology

- Controlled Vocabulary: One Thing Leads to Another[2]
- IA Institute[3]
- Lexonomy: a Taxonomy Primer[4]
- Visual Vocabulary, by Jesse James Garrett[5]

[2] http://www.controlledvocabulary.com
[3] http://www.iainstitute.org/en/learn/
[4] http://www.ischool.utexas.edu/~i385e/readings/Warner-aTaxonomyPrimer.html
[5] http://www.jjg.net/ia/visvocab/

www.it-ebooks.info

Navigation

Wayfinding is one of the most important elements of a website. Users expect navigation bars as they represent the most efficient way to go from category to category. Much has been said about the placement of navigation and elements. Should a navigation bar use images or drop-downs? Should it be standard, or customized to specific types of users? Essentially, a well-placed navigation bar with recognizable terms (such as Home, About Us, Contact Us, and so on) provides an optimal experience to the user. Navigation bars are typically seen on each page of a website, but some newer techniques use guided or faceted navigation, smart navigation metadata, and sliders. Unfortunately, some of these tools, especially those utilizing JavaScript and Ajax, have posed accessibility issues yet to be resolved (see the discussion in Chuck Longanecker's "What to Expect in 2010: UX/UI Design Simplicity" article[6] and the comments).

Breadcrumbs and similar user-location tools are also acceptable wayfinding techniques, with users coming to expect this visual trail at the top of each web page. Indeed, as author Marta Eleniak states in "Essential Navigation Checklists for Web Design,"[7] the "You Are Here" reference is especially useful during a web user interaction. Users want to see which step they're at (and how long they have to complete the transaction) when making a purchase or a hotel reservation.

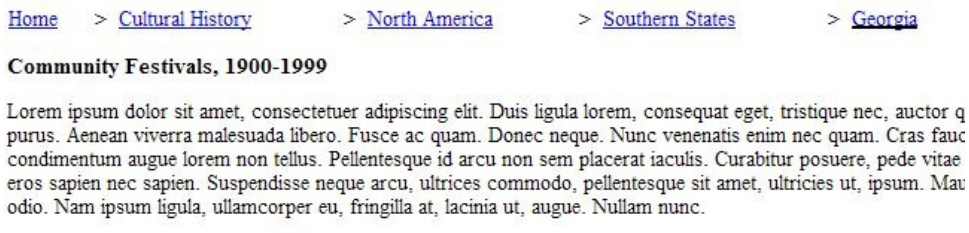
Home > Cultural History > North America > Southern States > Georgia
Community Festivals, 1900-1999
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis ligula lorem, consequat eget, tristique nec, auctor q
purus. Aenean viverra malesuada libero. Fusce ac quam. Donec neque. Nunc venenatis enim nec quam. Cras fauc
condimentum augue lorem non tellus. Pellentesque id arcu non sem placerat iaculis. Curabitur posuere, pede vitae
eros sapien nec sapien. Suspendisse neque arcu, ultrices commodo, pellentesque sit amet, ultricies ut, ipsum. Mau
odio. Nam ipsum ligula, ullamcorper eu, fringilla at, lacinia ut, augue. Nullam nunc.

Figure 8.2. An example of breadcrumb links with a "You Are Here" indicator

Footers have become more important in the information they carry. While most websites utilize generic information such as contact or copyright, others include more in-depth links to their sites, or even modified sitemaps. Yet, it's important

[6] http://www.webdesignerwall.com/trends/what-to-expect-in-2010-uxui-design-simplicity/
[7] http://articles.sitepoint.com/article/checklists-web-design/

www.it-ebooks.info

not to overload the footer area with content found elsewhere on the site—after all, why take up more valuable real estate?

Other Navigation Tools

Users may still require a little help navigating a site, despite the best efforts of the developer. To assist the user, the website may include such tools as dynamic databases, tags, faceted searching, images, search boxes, and sitemaps. A sitemap may be placed in a separate document on the site or discretely placed on each page. The search box should be placed prominently at the top of the page where most users expect to find it. Remember, though, that navigation tools should be kept to a minimum and not distract the user, who visits for content, not design. Text-only alternatives to the website should be considered to enhance accessibility.

Dead Ends

Error 404 "File Not Found" codes should be avoided as much as possible, since they can turn off a user (see Oscar Trelles' article, "'Not Found' Is Not An Option: Error Handling and User Experience"[8]). Instead, the user should receive a friendly, meaningful error message, as Trelles recommends, "Write the message in language that exhibits a level of familiarity that's appropriate for your site." Make it simple, but clear, and offer a way out, whether it's through a search box, a sitemap, or a link back to the home page.

Before Launching

There are a number of boxes you should tick before launching a new or redesigned site. Here are a few of the most important:

Usability

Usability testing is recommended before launching the website. This measures ease and efficiency of use, accessibility, error frequency, ease of interpretation, and general user satisfaction, among other things. Many types of usability tests are available, from eye-tracking studies to surveys. Rudimentary mockups called wireframes are frequently used by information architects to determine structure and user feedback. Usability is a complex field which warrants close investigation.

[8] http://articles.sitepoint.com/article/error-handling-user-experience

www.it-ebooks.info

Selected Resources on Usability

- Steve Krug's website: Advanced Common Sense[9] (author of *Don't Make Me Think!* and *Rocket Surgery Made Easy*)
- SitePoint's Design and Layout: Usability and Information Architecture[10] category
- Usability.gov[11]
- Usability Professionals' Association[12]
- Works by Jakob Nielsen[13]

Review of Textual Content

Information should be presented properly in all respects: design, content, and grammatical correctness. Developers and stakeholders should review their websites for grammatical errors, spelling mistakes, wordiness, writing style, and other language usage elements. Regrettably, there seems to be very little written on the subject of correct grammar and style usage for the Web. As a former English major and column editor of a state library journal, I am excessively conscious of grammatical and writing style problems. Most large websites appear to be more conscious of such issues, but all businesses should try to pay more attention to their grammatical content. Abundant grammatical indiscretions may signal that a site is unprofessional, and drive users away.

A Few Recommended Grammatical Usage Sites

- GrammarErrors.com[14]
- Grammar Girl: Quick and Dirty Tips for Better Writing[15]

[9] http://www.sensible.com/
[10] http://articles.sitepoint.com/category/usability
[11] http://usability.gov/
[12] http://www.upassoc.org/
[13] http://www.useit.com/
[14] http://www.grammarerrors.com/
[15] http://grammar.quickanddirtytips.com

www.it-ebooks.info

- Purdue University Online Writing Lab (OWL)[16]

Miscellaneous Information

Before moving on to the design element, other information elements that need to be considered include page titles, contact emails, sitemap files, keywords, and metadata description of the content, as they are helpful for user searching and search engine crawling. These simple elements are sometimes overlooked by the developer, especially when a site is updated.

The End is (Never) Near

Information organization is just one factor of a successful website. Combined with a consistent and clean design that respects both the users and the website's stakeholders, a developer will be able to achieve a high level of design and functionality. Resources such as Jason Beaird's *The Principles of Beautiful Web Design*[17] provide essential information on style and design. The dynamic nature of the Web, technological innovations, and frequent changes in information content represent many challenges, but with a sense of humor and an enthusiasm for change, the developer will no doubt achieve success.

[16] http://owl.english.purdue.edu/
[17] http://sitepoint.com/books/design2/

www.it-ebooks.info

www.it-ebooks.info

Chapter 9

Using Vector Graphics to Build a Noughts & Crosses Game

by Clive Wickham

This tutorial outlines how you can easily build a simple game of Noughts & Crosses using interactive vector graphics. We'll explore two ways of presenting our game on the Web—SVG and (as an alternative) HTML5 Canvas—comparing them as we go along. Be prepared to get your hands dirty, as we'll be using tubes of JavaScript to glue each of the two examples together. Amazingly, there'll be no need to open any image editing software to achieve all of the above!

What is Noughts and Crosses?

For the uninitiated, Noughts & Crosses is a simple game from my childhood. Traditionally played using a pen and paper as shown in Figure 9.1, it's a two-player game that typically uses a 3×3 grid. Each player is assigned either a nought (O) or a cross (X), where they then take turns to place their symbol in an empty square on the grid. The winner is the first player to place three noughts (or three crosses) in con-

www.it-ebooks.info

secutive squares of the grid in a vertical, horizontal, or diagonal direction. North Americans will know it as tic-tac-toe.

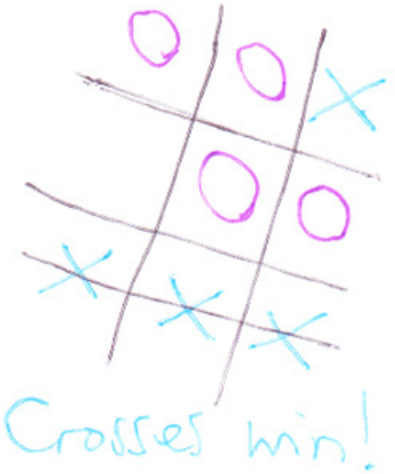

Figure 9.1. A noughts & crosses game—the traditional way

Vector Graphics

There are basically two types of image formats out there in the wilds of the Web:

- Pixel-based **raster** graphics (nothing to do with dreadlocks!), which represent an image as a grid, or bitmap, of small coloured squares called **pixels**. Raster graphics are bound to a specific resolution, which means the image is stretched when viewed at a different resolution. At low resolutions, pixelation may occur where individual pixels are visible to the eye. This category is generally better for photographs, and encompasses the very familiar BMP, JPG, GIF, and PNG file formats.

- Line-based **vector** graphics, which represent an image as a series of instructions to draw primitive shapes such as lines, rectangles, and circles. Vector graphics are independent of resolution, so the instructions stored in the file are used to determine how to draw the image smoothly at any resolution. Vector graphics are more suitable for the display of logos, illustrations, and images with text. This category includes the less familiar SVG, VML, and EMF file formats.

www.it-ebooks.info

The main advantage of vector images over raster images is that they can scale to any size without loss of quality. This is particularly useful when considering the vast array of screen sizes used to browse the Web—from tiny mobile device screens to massive workstation or television screens. You can see where the name Scalable Vector Graphics (SVG) came from.

Vector Graphics with SVG

SVG is an open standard for defining static, animated, and interactive vector graphics for the Web. As a textual language, SVG can be readily compressed, indexed by search engines, and easily generated by server-side languages. Client-side scripting languages can also interact with SVG documents to bring about dynamic behavior. To give you a flavor of what SVG markup looks like, here's an example of a very simple SVG document that draws a rectangle with a width of 50 units and a height of 100 units:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="50" height="100" />
</svg>
```

When this simple SVG document is opened in a browser, you'll see a black rectangle (like the one in Figure 9.2), because we've not specified a fill color using the fill attribute so the default black color is used.

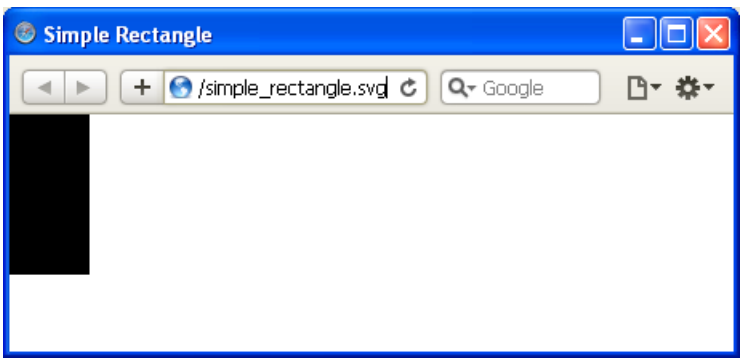

Figure 9.2. A simple SVG document

You might be thinking that the opening `svg` tag looks messy. It does, but it's necessary. SVG is a variety of XML, and web browsers need to know this to interpret and display SVG documents correctly. The `xmlns` attribute in the code above represents

www.it-ebooks.info

a grouping or category within XML called an XML namespace; it tells the browser to treat the XML elements within the document as the SVG variety of XML.

Basic SVG support is available in the following browser versions: Firefox 3.0+, Safari 3.2+, Chrome 4.0+, and Opera 10.1+. Unfortunately, Internet Explorer 8 (IE8) and earlier versions do not natively support SVG; however, basic SVG functionality is provided in the IE9 beta (a future version of IE yet to be released to the general public). In the meantime, the Adobe SVG Viewer (ASV) plugin can be downloaded and used in IE to view and interact with SVG content.

At the time of writing, SVG support in Google Chrome (version 6.0.472.63) is not quite ready for prime time:

- A reference to an external SVG document is unavailable (using the `getSVGDocument` method) when running from the local file system, but functions correctly when executed from a web server.
- Elements added dynamically to an external SVG document (using the code in the sections that follow) do not show—as if some kind of refresh is required on the SVG canvas; however, these elements do appear once Chrome's Zoom feature has been used or the browser window has been resized.

Vector Graphics with HTML5 Canvas

Currently under development as a working draft, HTML5 is the next major version of HTML. It includes a new `canvas` element that provides a raster/bitmap canvas for dynamically drawing images using a scripting language. Adding a blank `canvas` element into your HTML5 document is straightforward:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Rectangle</title>
  </head>
  <body>
    <canvas id="theCanvas" height="200" width="200">
      Fallback content displayed when canvas is not supported
    </canvas>
  </body>
</html>
```

www.it-ebooks.info

The Canvas 2D API is a set of JavaScript methods that can be used to draw two-dimensional vector graphics on the `canvas` element. For example, to reproduce the black rectangle we drew earlier using SVG:

```
<canvas id="theCanvas" height="200" width="200"></canvas>
<script type="text/javascript">
  var canvas = document.getElementById('theCanvas');
  var context = canvas.getContext('2d');
  context.fillRect(0, 0, 50, 100);
</script>
```

Figure 9.3 shows the output in a web browser.

Figure 9.3. A simple HTML5 document using the canvas element

The first line finds the `canvas` element within the Document Object Model (DOM) using the `document.getElementById` method. To actually draw on the `canvas` element we've located, we need to first get the 2D drawing context of the canvas by calling the `getContext('2d')` method. At the time of writing, only the two-dimensional context has been defined. This drawing context provides access to all the drawing methods that can be applied to the canvas. A good analogy would be to think of the `canvas` element as an artist's canvas and the drawing context as an artist's tools (for example, paints, brushes, and crayons) and actions. So, the canvas element is simply a container for graphics, and JavaScript actually draws the graphics.

Basic support for the `canvas` element (including the ability to display text on `canvas` elements) is available in the following browser versions: Firefox 3.5+, Safari 4.0+, Chrome 4.0+, and Opera 10.5+. Unfortunately, IE8 and earlier versions do not nat-

www.it-ebooks.info

ively support the canvas element; however, it looks like it may well be supported in the future—in IE9.[1] In the meantime, a third-party script called ExplorerCanvas[2] (which brings canvas support to IE) has fortunately been made available, and can easily be integrated into existing web pages using a single script tag within conditional comments:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Web Page Title</title>
  <!--[if IE]>
    <script src="ExplorerCanvas\excanvas.js"></script>
  <![endif]-->
```

In case IE9 does include native support for the canvas element, it's easy to modify the above conditional comment to only target IE browsers released prior to IE9 (lte in the code below stands for "less than or equal to"):

```
  <!--[if lte IE 8]>
    <script src="ExplorerCanvas\excanvas.js"></script>
  <![endif]-->
```

Building a Simple Game of Noughts & Crosses

Now onto the real meat of the tutorial: building our simple game of Noughts & Crosses in SVG and HTML5 Canvas. We'll need to combine a variety of technologies for each example and, just so that you're clear, the purpose of each technology is listed in the table following below.

[1] http://blogs.msdn.com/b/ie/archive/2010/03/09/working-with-the-html5-community.aspx
[2] http://code.google.com/p/explorercanvas/

www.it-ebooks.info

Table 9.1. SVG example

Technology	Purpose	Filename
HTML5	■ to create the web document in which the SVG document is embedded ■ to explain the rules of the game ■ to provide a button for playing the game again	**noughts_and_crosses_svg.htm**
SVG	■ to provide an area on which to draw the output of the game ■ to define the various game elements that will be drawn	**noughts_and_crosses.svg**
JavaScript	■ to make the SVG area clickable ■ to respond to user mouse clicks ■ to draw the various components of the game ■ to implement the game logic	**noughts_and_crosses_svg.js**
CSS	■ to style the text of the HTML document	**styles.css**

www.it-ebooks.info

Table 9.2. HTML5 canvas example

Technology	Purpose	Filename
HTML5	■ to create the web document in which the canvas element resides ■ to provide a canvas on which to draw the output of the game ■ to explain the rules of the game ■ to provide a button for playing the game again	**noughts_and_crosses_canvas.htm**
JavaScript	■ to make the HTML5 canvas clickable ■ to respond to user mouse clicks ■ to draw the various components of the game ■ to implement the game logic	**noughts_and_crosses_canvas.js**
CSS	■ to style the text of the HTML document	**styles.css**

Indispensable JavaScript

JavaScript—not to be confused with the Java programming language—is the most popular client-side scripting language on the Internet, having been available in web browsers since 1996. JavaScript can be used for controlling the behavior or interactivity of web-based documents via the DOM.

The key technology for driving both vector technologies is JavaScript. Without JavaScript, nothing can be done with the HTML5 canvas—it will remain blank. In SVG, too, we require JavaScript to construct our game graphics by dynamically creating, cloning, and configuring SVG elements.

Creating an HTML Template

First up, we need a web page in which to display our Noughts & Crosses game. We'll use HTML5 syntax to form our page, so here's our basic template:

www.it-ebooks.info

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Noughts &amp; Crosses Game</title>
  <link rel="stylesheet" type="text/css" href="styles.css" />
  <script type="text/javascript" src="filename_here.js"></script>
</head>
<body>
<h1>Play "Noughts &amp; Crosses"</h1>
<p>Blurb about the game...</p>
<!-- ------------------------------- -->
<!-- SVG or HTML5 Canvas will go here -->
<!-- ------------------------------- -->
<p>
  Player 1: <span id="noughtsText">Noughts</span>
  Player 2: <span id="crossesText">Crosses</span>
</p>
<p><button id="newGameButton">Play Again</button></p>
</body>
</html>
```

Some points to note about our HTML template:

- We've included some simple CSS styling of our HTML content by linking to an external CSS file.
- We've linked to an external JavaScript file so as to be unobtrusive and not clutter our HTML file, and the following values should be substituted for `filename_here.js`:
  - for SVG: `noughts_and_crosses_svg.js`
  - for HTML5 Canvas: `noughts_and_crosses_canvas.js`
- Two `span` elements are used to display text in the different colors used to represent the noughts and the crosses.
- A button is used to start a new game.

www.it-ebooks.info

Embedding an External SVG Document

To embed our external SVG document in our HTML template, we need to use an `object` element like so:

noughts_and_crosses_svg.html *(excerpt)*

```
<object id="gameCanvas" data="noughts_and_crosses.svg"
    type="image/svg+xml" height="300" width="500">
  <!--[if IE]>
    <param name="src" value="noughts_and_crosses.svg">
  <![endif]-->
  <p>Content to display if browser offers no SVG support</p>
</object>
```

All modern browsers, except IE, are happy to display SVG using the `object` element. IE requires an additional `param` element in order to get interactivity working, so we've included the `param` element in a conditional comment so that browsers other than IE will ignore it.

Inserting an HTML5 canvas Element

The HTML5 `canvas` element is inserted in our alternative HTML template as follows:

noughts_and_crosses_canvas.htm *(excerpt)*

```
<canvas id="gameCanvas" height="300" width="500">
  <p>Content to display if browser offers no Canvas support</p>
</canvas>
```

Creating the SVG Blueprint

Now that we have an area to draw on for each example, we can start preparing the blueprint for the building blocks that we'll need to add. Let's start with creating our SVG document:

noughts_and_crosses.svg *(excerpt)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-flat-20030114.dtd">
<svg xmlns="http://www.w3.org/2000/svg" height="300" width="300">
```

www.it-ebooks.info

```
  <title>Noughts and Crosses Game</title>
  <defs>
    <!-- vertical grid line -->
    <!-- horizontal grid line -->
    <!-- grid -->
    <!-- nought -->
    <!-- cross -->
    <!-- strikethrough -->
    <!-- gradient for game over background -->
    <!-- game over notice -->
  </defs>
  <g id="gameContainer"></g>
</svg>
```

Prior to the `svg` element, there are several items we need to declare. First, we have to declare the document as being an XML document (the first line). Second, we have to declare that the document type is the SVG variety of XML (the second and third lines). Next comes the familiar `svg` element with dimensions of 300×300px units. The `title` element is self-explanatory. The `defs` element is where all our building block elements will be defined. Comments have been included in this section to indicate each of the building blocks we'll define. Anything within a `defs` element will not be a visible part of the document; however, we'll later use these definitions in a group (`g`) element called `gameContainer`, which will act as the container for all the elements we want to dynamically add (and be visible), and will make it easy for us to clear any elements we've added when we come to start a new game.

Defining the Building Blocks for Our Game

Let's define the playing grid first. In our SVG document, we'll need to define one vertical line element and one horizontal line element, which we'll reuse to form the grid:

noughts_and_crosses.svg *(excerpt)*

```
<!-- vertical grid line -->
<line id="verticalGridLine" x1="0" y1="0" x2="0" y2="300" />
<!-- horizontal grid line -->
<line id="horizontalGridLine" x1="0" y1="0" x2="300" y2="0" />
```

www.it-ebooks.info

When drawn, our vertical grid line will therefore stretch from the top left of the canvas (0, 0) to the bottom left (0, 300). Our horizontal grid line will stretch from the top left of the canvas (0, 0) to the top right (300, 0). Next, we'll define our grid using the above line elements and position them exactly where we want them to be:

noughts_and_crosses.svg *(excerpt)*

```
<!-- grid -->
<g id="grid" stroke-width="7" fill="none">
  <use xlink:href="#verticalGridLine" x="100" y="0" />
  <use xlink:href="#verticalGridLine" x="200" y="0" />
  <use xlink:href="#horizontalGridLine" x="0" y="100" />
  <use xlink:href="#horizontalGridLine" x="0" y="200" />
</g>
```

We've used the `g` element to define a group (our grid) made up of four reused line elements. The grid won't have a background color as the `fill` attribute is set to `none`. A thick line will be used to draw the lines as a result of setting `stroke-width` to `7`. To reuse predefined elements, we must employ the `use` element and refer to the predefined line elements by using the `xlink:href` attribute. XLink is the XML Linking Language, and we need to use it to link to elements we've predefined. For this to happen, we'll have to include an additional XML namespace in our `svg` element so that features of XLink can be used throughout the document:

noughts_and_crosses.svg *(excerpt)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-flat-20030114.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     height="300" width="300">
```

Within the grid `g` element, the two vertical lines are repositioned one-third of the way and two-thirds of the way across the canvas. The two horizontal lines are repositioned one-third of the way and two-thirds of the way down the canvas.

This step of defining our grid is unnecessary for our HTML5 `canvas` example because the `canvas` element doesn't intrinsically use the same concept of elements or objects (although this behavior could be simulated by utilizing objects within JavaScript),

www.it-ebooks.info

and therefore has no inherent ability to predefine, and later use and reuse, elements. For our example, we'll simply issue commands to draw straight onto the canvas.

Drawing the Playing Grid

In our JavaScript file for the SVG example, we need to create a function we can later call to draw the grid—let's call it `drawGrid`. What this function should do is dynamically create a new `use` element, which references the grid group element we defined above, and set appropriate attributes for that `use` element:

noughts_and_crosses_svg.js *(excerpt)*

```
var COLOUR_GRID = 'silver';  // colour of grid lines
var SVG_NS = 'http://www.w3.org/2000/svg';      // SVG namespace
var XLINK_NS = 'http://www.w3.org/1999/xlink'; // XLink namespace

function drawGrid(){
  var newUseElement = gSvgDoc.createElementNS(SVG_NS, 'use');
  newUseElement.setAttributeNS(null, 'stroke', COLOUR_GRID);
  newUseElement.setAttributeNS(null, 'x', 0);
  newUseElement.setAttributeNS(null, 'y', 0);
  newUseElement.setAttributeNS(XLINK_NS, 'href', '#grid');
  gGameContainer.appendChild(newUseElement);
}
```

You'll notice we've also created three global variables; the latter two are for use throughout the script. `COLOUR_GRID` represents a constant value, which we'll keep at the top of our JavaScript file, so that the colors of the game's component parts can be easily modified without hunting through the code. The `drawGrid` function above will dynamically create the following SVG `use` element and insert it into the previously mentioned `gameContainer` group element:

```
<g id="gameContainer">
  <use stroke="silver" x="0" y="0" xlink:href="#grid" />
</g>
```

This code will give the playing grid lines a silver color, as shown in Figure 9.4, and position it in the top-left of our SVG document.

www.it-ebooks.info

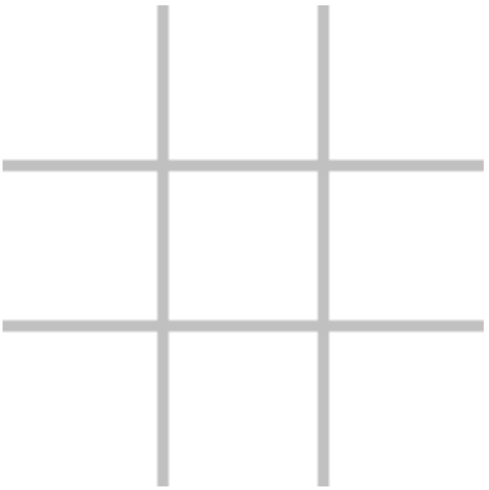

Figure 9.4. The playing grid in SVG

In our JavaScript file for the HTML5 canvas example, we also need a `drawGrid` function, but it will need to cause some different behavior. What this function should do is dynamically draw the lines that make up the grid, equivalent to the SVG version in Figure 9.4:

noughts_and_crosses_canvas.js *(excerpt)*

```
var COLOUR_GRID = 'silver';  // colour of grid lines

function drawGrid() {
  // configure grid lines
  gContext.lineWidth = LINE_WIDTH_MULTIPLIER * gGridWidth;
  gContext.strokeStyle = COLOUR_GRID;
  gContext.beginPath();
  // draw vertical grid lines
  gContext.moveTo(gCellWidth, 0);
  gContext.lineTo(gCellWidth, gGridHeight);
  gContext.moveTo(gCellWidth * 2, 0);
  gContext.lineTo(gCellWidth * 2, gGridHeight);
  // draw horizontal grid lines
  gContext.moveTo(0, gCellHeight);
  gContext.lineTo(gGridWidth, gCellHeight);
  gContext.moveTo(0, gCellHeight * 2);
  gContext.lineTo(gGridWidth, gCellHeight * 2);
  gContext.closePath();
```

www.it-ebooks.info

```
  // make lines permanent
  gContext.stroke();
}
```

You'll notice several additional global variables are used in this function:

- `gGridWidth` is the width of the playing grid
- `gGridHeight` is the height of the playing grid
- `gCellWidth` is the width of a single cell/square in the grid
- `gCellHeight` is the height of a single cell/square in the grid
- `LINE_WIDTH_MULTIPLIER` is used to set the grid line width in proportion to the width of the playing grid

Before drawing our grid, we set up the `lineWidth` and `strokeStyle` properties of the drawing context. These are equivalent to the `stroke-width` and `stroke` attributes we saw in SVG. We then call the `beginPath` method to commence the drawing of our grid. We use the `moveTo(x,y)` and `lineTo(x,y)` methods to draw two vertical lines positioned one-third of the way (`gCellWidth`) and two-thirds of the way (`gCellWidth × 2`) across the canvas, stretching from the top of the canvas (y = 0) to the bottom of the canvas (y = `gGridHeight`). The `moveTo` method is like lifting your pencil up and moving it into position, and the `lineTo` method is like applying the pencil to the canvas and drawing to a certain position. Similarly, the same methods are used to draw two horizontal lines positioned one-third of the way (`gCellHeight`) and two-thirds of the way (`gCellHeight × 2`) down the canvas, and stretching from the left of the canvas (x = 0) to the right of the canvas (x = `gGridWidth`). The `closePath` method is called to end the drawing of our grid. To make the lines we've drawn permanent—imagine you're drawing over the pencil lines with an ink pen—we use the `stroke` method and, hey presto, our `drawGrid` function is ready to draw our playing grid when a call to it is made.

Defining the SVG Nought

In our SVG document, we'll need to define a "nought" element, which will be placed on the grid when a grid cell is clicked:

www.it-ebooks.info

noughts_and_crosses.svg *(excerpt)*

```
<!-- nought -->
<circle id="nought" r="35" stroke-width="7" fill="none"
    cx="50" cy="50" />
```

Helpfully, there's a `circle` element in SVG that can be easily used to draw our nought. We specify a radius of 35 units using the `r` attribute. We want the nought to be drawn with an outline (that is, a stroke) and not filled in, so we set a `stroke-width` of 7 to match that of the grid (a color will be dynamically assigned to the `stroke` attribute in our JavaScript code) and a `fill` of `none`. The last two attributes, `cx` and `cy`, denote the position of the center of the circle.

Drawing a Nought

In our JavaScript file for the SVG example, we need to create a function we can call to draw a nought in a specific cell of the grid—let's call it `drawNought`. What this function should do is dynamically clone the circle element we defined above and set appropriate attributes for that element:

noughts_and_crosses_svg.js *(excerpt)*

```
var COLOUR_NOUGHT = 'blue';  // colour of noughts

function drawNought(cell) {
  var noughtTemplate = gSvgDoc.getElementById('nought');
  var clonedNought = noughtTemplate.cloneNode(false);
  // convert column & row of cell to coords for "nought" centre
  var x = (gCellWidth * cell.column) + (gCellWidth / 2);
  var y = (gCellHeight * cell.row) + (gCellHeight / 2);
  clonedNought.removeAttribute('id');
  clonedNought.setAttributeNS(null, 'stroke', COLOUR_NOUGHT);
  clonedNought.setAttributeNS(null, 'cx', x);
  clonedNought.setAttributeNS(null, 'cy', y);
  gGameContainer.appendChild(clonedNought);
  // record turn taken
  gTurns[cell.column][cell.row] = 'O';
}
```

We first use the `getElementById` function to find our circle element within the SVG document using its `id` attribute of `nought`. We store the value it returns in a variable called `noughtTemplate`. We can then call the `cloneNode` method of `noughtTemplate`

www.it-ebooks.info

to make a copy of our original circle element, passing an argument of `false` because we don't want to clone any child nodes of the `circle` element (actually, there aren't any child nodes within the `circle` element so it wouldn't do any harm to pass `true`!). A *`cell`* argument is passed into the `drawNought` function, which consists of a `column` property and a `row` property. These properties are used in conjunction with `gCellWidth` and `gCellHeight` to calculate the corresponding coordinates of the center of the cell in which the cloned nought is to be drawn. The `cx` and `cy` attributes of the cloned nought are then set accordingly with the calculated coordinates, and its outline (stroke) color is set to blue (as per the value assigned to `COLOUR_NOUGHT`). This cloned nought is then inserted into the previously mentioned `gameContainer` group element. To help us keep track of where on the grid noughts and crosses have been placed, we'll use a two-dimensional array called `gTurns`, and will assign a `O` or `X` character to an array item that corresponds to the column and row of the appropriate cell.

The `drawNought` function above will dynamically create an SVG circle element and insert it into the previously mentioned `gameContainer` group element like so:

```
<g id="gameContainer">
  <use stroke="silver" x="0" y="0" xlink:href="#grid" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none"
      cx="250" cy="250" />
</g>
```

This SVG code will draw a blue nought in the bottom-right cell of the playing grid, as in Figure 9.5.

www.it-ebooks.info

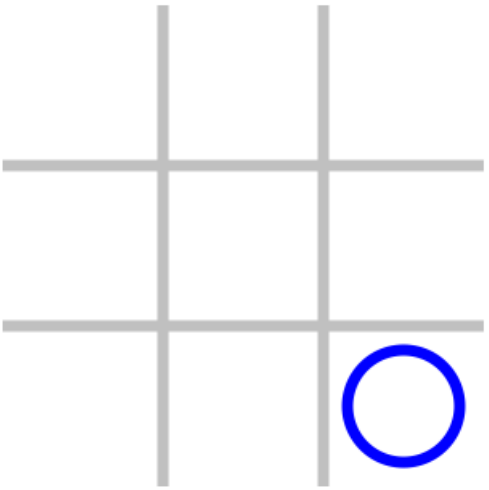

Figure 9.5. A nought on the playing grid in SVG

In our JavaScript file for the HTML5 canvas example, we also need a `drawNought` function, but it needs to behave a little differently. What this function should do is dynamically draw a nought in an appropriate cell of the grid, equivalent to the SVG version in Figure 5:

noughts_and_crosses_canvas.js *(excerpt)*

```
var COLOUR_NOUGHT = 'blue';  // colour of noughts

function drawNought(cell) {
  // radius of arc to be drawn to represent a "nought",
  // x & y coordinates for arc centre point
  var radius, x, y;
  // convert column and row of cell to coordinates
  x = (gCellWidth * cell.column) + (gCellWidth / 2);
  y = (gCellHeight * cell.row) + (gCellHeight / 2);
  // set radius of nought such that it will fill a cell,
  // taking cell padding into account
  radius = (gCellWidth / 2) - gCellPadding;
  // configure line used to draw the nought
  gContext.lineWidth = LINE_WIDTH_MULTIPLIER * gGridWidth;
  gContext.strokeStyle = COLOUR_NOUGHT;
  gContext.beginPath();
  // draw circle to represent the nought
  gContext.arc(x, y, radius, 0, Math.PI * 2, false);
  gContext.closePath();
```

www.it-ebooks.info

```
  // make circle permanent
  gContext.stroke();
  // record turn taken
  gTurns[cell.column][cell.row] = 'O';
}
```

Before drawing our nought, we convert the `column` and `row` properties of the *`cell`* argument (that is passed into the function) into x and y coordinates representing the center of the appropriate cell. We calculate that the radius of the nought should be half the width of a cell minus a bit of padding (`gCellPadding`) to ensure a gap between the nought and the cell edges. As before, we set up the `lineWidth` and `strokeStyle` properties to match what we did in the SVG version. We then call the `beginPath` method to start the drawing of our nought. We then use the `arc` method to draw our nought, passing the following to it: the x and y coordinates for the center of the nought, the radius of the nought, the starting angle of 0 radians (which is equivalent to 0 degrees), the ending angle of 2π radians (which is equivalent to 360 degrees) going in a clockwise direction (indicated by the final `false` argument. Fortunately, we can use JavaScript's built-in `Math` module to get the exact value of pi and thus help us to calculate radian angle values, a unit that many may be unfamiliar with. The `closePath` method is called to end the drawing of our nought. We call the `stroke` method to make the nought permanent, and then record the cell in which the nought was drawn by setting the appropriate item of the `gTurns` array, as above.

Defining the SVG Cross

In our SVG document, we'll need to define a "cross" element, which will be placed on the grid when a grid cell is clicked. We could do this using two `line` elements within a group element like so:

```
<!-- cross -->
<g id="cross" stroke-width="7" fill="none">
  <line x1="0" y1="0" x2="70" y2="70" />
  <line x1="70" y1="0" x2="0" y2="70" />
</g>
```

Or better still, we can create the same output using a single path element:

www.it-ebooks.info

noughts_and_crosses.svg *(excerpt)*

```
<!-- cross -->
<path id="cross" d="M 0,0 L 70,70 M 70,0 L 0,70"
    stroke-width="7" fill="none" />
```

Although it looks a bit confusing, the `d` attribute represents path data and defines the outline of a shape in a really concise way. It's easy to decipher when you know what the `M` and `L` characters represent. `M` indicates a "move to" instruction (that is, moving to a position with the pen up) and `L` indicates a "line to" instruction (moving to a position with the pen down). So the value assigned to the `d` attribute above is a concise way of expressing the following set of instructions:

- Lift the pen up from the canvas, and move it to 0,0
- Put the pen down on the canvas, and move it to 70,70
- Lift the pen up from the canvas, and move it to 70,0
- Put the pen down on the canvas, and move it to 0,70

As before, we want the cross to be drawn with an outline (a stroke) but not filled in, so we set a `stroke-width` of `7` to match that of the nought (a color will be dynamically assigned to the `stroke` attribute in our JavaScript code) and a `fill` of `none`.

Drawing a Cross

In our JavaScript file for the SVG example, we need to create a function we can call to draw a cross in a specific cell of the grid—let's call it `drawCross`. What this function should do is dynamically create a new `use` element, which references the cross `path` element we defined above, and set appropriate attributes for that use element:

noughts_and_crosses_svg.js *(excerpt)*

```
var COLOUR_CROSS = 'red';  // colour of crosses

function drawCross(cell) {
  var newUseElement = gSvgDoc.createElementNS(SVG_NS, 'use');
  var x = (gCellWidth * cell.column) + gCellPadding;
  var y = (gCellHeight * cell.row) + gCellPadding;
  newUseElement.setAttributeNS(null, 'stroke', COLOUR_CROSS);
  newUseElement.setAttributeNS(null, 'x', x);
  newUseElement.setAttributeNS(null, 'y', y);
```

www.it-ebooks.info

```
  newUseElement.setAttributeNS(XLINK_NS, 'href', '#cross');
  gGameContainer.appendChild(newUseElement);
  // record turn taken
  gTurns[cell.column][cell.row] = 'X';
}
```

The `drawCross` function above will dynamically create the following SVG `use` element and insert it into the previously mentioned `gameContainer` group element, positioning it using coordinates calculated from the column and row properties passed to the function via the *`cell`* argument and outlining it in red (as per the value assigned to `COLOUR_CROSS`):

```
<g id="gameContainer">
  <use stroke="silver" x="0" y="0" xlink:href="#grid" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none"
      cx="250" cy="250" />
  <use stroke="red" x="215" y="15" xlink:href="#cross" />
</g>
```

This SVG code will draw a red cross in the top right cell of the playing grid and a blue nought in the bottom right cell, as in Figure 9.6.

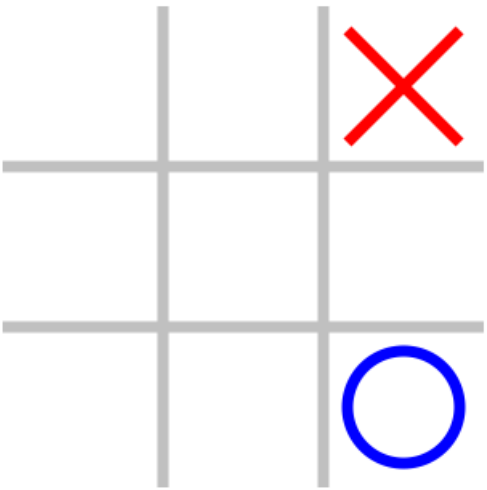

Figure 9.6. A cross on the playing grid in SVG

In our JavaScript file for the HTML5 canvas example, we also need a `drawCross` function, and it will cause very similar behavior to the SVG `path` element. What

www.it-ebooks.info

this function should do is dynamically draw two diagonal lines to represent a cross in an appropriate cell of the grid, equivalent to the SVG version in Figure 9.6:

noughts_and_crosses_canvas.js *(excerpt)*

```
var COLOUR_CROSS = 'red';  // colour of crosses

function drawCross(cell) {
  // configure line used to draw the cross
  gContext.lineWidth = LINE_WIDTH_MULTIPLIER * gGridWidth;
  gContext.strokeStyle = COLOUR_CROSS;
  gContext.beginPath();
  // draw two diagonal lines to represent cross
  gContext.moveTo((gCellWidth * cell.column) + gCellPadding,
                  (gCellHeight * cell.row) + gCellPadding);
  gContext.lineTo((gCellWidth * (cell.column + 1)) - gCellPadding,
                  (gCellHeight * (cell.row + 1)) - gCellPadding);
  gContext.moveTo((gCellWidth * (cell.column + 1)) - gCellPadding,
                  (gCellHeight * cell.row) + gCellPadding);
  gContext.lineTo((gCellWidth * cell.column) + gCellPadding,
                  (gCellHeight * (cell.row + 1)) - gCellPadding);
  gContext.closePath();
  // make lines permanent
  gContext.stroke();
  // record turn taken
  gTurns[cell.column][cell.row] = 'X';
}
```

Before drawing our cross, we set up the `lineWidth` and `strokeStyle` properties to match what we did in the SVG version. We then call the `beginPath` method to start the drawing of our cross. We use a combination of `moveTo` and `lineTo` calls to draw the first diagonal line from the top left to the bottom right of the appropriate cell (calculating the appropriate x and y coordinates as we pass them as arguments to the `moveTo` and `lineTo` methods). We use the `moveTo` and `lineTo` combination again to draw the second diagonal line from the top right to the bottom left of the appropriate cell. This is the same kind of process we used in the SVG `path` element. The `closePath` method is called to end the drawing of our cross. We call the `stroke` method to make the cross permanent, and then record the cell in which the cross was drawn by setting the appropriate item of the `gTurns` array, as above.

www.it-ebooks.info

Defining the SVG Strikethrough

In our SVG document, we'll need to define a "strikethrough" element, which will be placed on the grid to highlight three noughts, or three crosses—that is, a winning streak—in a line:

noughts_and_crosses.svg *(excerpt)*

```
<!-- strikethrough -->
<line id="strikethrough" x1="50" y1="50" x2="50" y2="250"
    stroke-width="7" fill="none" />
```

Our strikethrough is simply a `line` element that will be drawn from x1,y1 to x2,y2 with a `stroke-width` matching our grid, noughts, and crosses. The x1, y1, x2, and y2 attributes will be reset dynamically using JavaScript so as to draw a vertical, horizontal, or diagonal line through three adjacent noughts or three adjacent crosses as appropriate. JavaScript will also be used to assign a color by setting the `stroke` attribute.

Drawing a Strikethrough

In our JavaScript file for the SVG example, we need to create a function we can call to draw a strikethrough line through three adjacent noughts or three adjacent crosses—let's call it `drawStrikeThrough`. What this function should do is dynamically clone the `line` element we defined above and set appropriate attributes for that element:

noughts_and_crosses_svg.js *(excerpt)*

```
var COLOUR_STRIKETHROUGH = 'orange';  // colour of strikethrough

function drawStrikeThrough(startCell, endCell) {
  // x and y coordinates corresponding to
  // startCell and endCell arguments
  var endX, endY, startX, startY;
  var strikethroughTemplate =
➥gSvgDoc.getElementById('strikethrough');
  var clonedStrikethrough = strikethroughTemplate.cloneNode(false);
  // convert startCell and endCell column and row numbers to coords
  startX = (gCellWidth * startCell.column) + (gCellWidth / 2);
  startY = (gCellHeight * startCell.row) + (gCellHeight / 2);
  endX = (gCellWidth * endCell.column) + (gCellWidth / 2);
```

www.it-ebooks.info

```
  endY = (gCellHeight * endCell.row) + (gCellHeight / 2);
  clonedStrikethrough.removeAttribute('id');
  clonedStrikethrough.setAttributeNS(null, 'stroke',
➥COLOUR_STRIKETHROUGH);
  clonedStrikethrough.setAttributeNS(null, 'x1', startX);
  clonedStrikethrough.setAttributeNS(null, 'y1', startY);
  clonedStrikethrough.setAttributeNS(null, 'x2', endX);
  clonedStrikethrough.setAttributeNS(null, 'y2', endY);
  gGameContainer.appendChild(clonedStrikethrough);
}
```

We first use the `getElementById` function to find our `line` element within the SVG document using its `id` attribute of `strikethrough`. We store the value it returns in a variable called `strikethroughTemplate`. We can then call the `cloneNode` method of `strikethroughTemplate` to make a copy of our original `line` element. Two arguments, *`startCell`* and *`endCell`*, are passed into the `drawStrikeThrough` function, each consisting of a `column` property and a `row` property. These properties are used in conjunction with `gCellWidth` and `gCellHeight` to calculate the corresponding coordinates of the center of the cell in which the strikethrough line should start and the center of the cell in which it should end. The x1, y1, x2, and y2 attributes of the cloned strikethrough are then set accordingly with the calculated coordinates, and its outline (stroke) color is set to orange (as per the value assigned to `COLOUR_STRIKETHROUGH`). This cloned strikethrough is then inserted into the `gameContainer` group element:

```
<g id="gameContainer">
  <use stroke="silver" x="0" y="0" xlink:href="#grid" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none" cx="50"
      cy="50" />
  <use stroke="red" x="115" y="115" xlink:href="#cross" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none" cx="150"
      cy="50" />
  <use stroke="red" x="215" y="115" xlink:href="#cross" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none" cx="250"
      cy="50" />
  <line x1="50" y1="50" x2="250" y2="50" stroke="orange"
      stroke-width="7" fill="none" />
</g>
```

This SVG code will draw a strikethrough line through three noughts in the top row of the playing grid, as in Figure 9.7.

www.it-ebooks.info

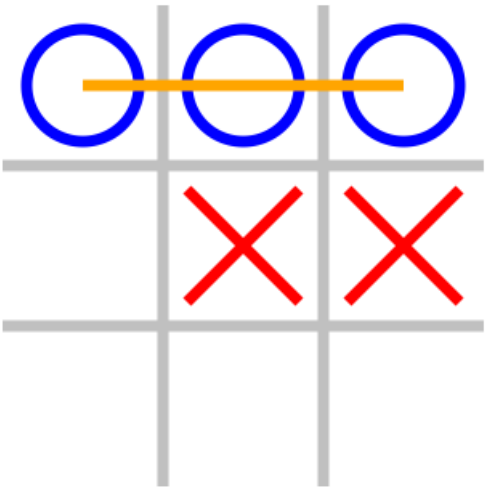

Figure 9.7. A strikethrough on the playing grid in SVG

In our JavaScript file for the HTML5 canvas example, we also need a `drawStrikeThrough` function. What this function should do is dynamically draw a line from the center of one cell to the center of another cell two cells away in a horizontal, vertical, or diagonal direction, equivalent to the SVG version in Figure 9.7:

noughts_and_crosses_canvas.js *(excerpt)*

```
var COLOUR_STRIKETHROUGH = 'orange';  // colour of strikethrough

function drawStrikeThrough(startCell, endCell) {
  // x and y coordinates corresponding to
  // startCell and endCell arguments
  var endX, endY, startX, startY;
  // convert startCell and endCell column and
  // row numbers to coordinates
  startX = (gCellWidth * startCell.column) + (gCellWidth / 2);
  startY = (gCellHeight * startCell.row) + (gCellHeight / 2);
  endX = (gCellWidth * endCell.column) + (gCellWidth / 2);
  endY = (gCellHeight * endCell.row) + (gCellHeight / 2);
  // configure strikethrough line
  gContext.lineWidth = STRIKETHROUGH_LINE_WIDTH_MULTIPLIER *
gGridWidth;
  gContext.strokeStyle = COLOUR_STRIKETHROUGH;
  gContext.beginPath();
  // draw strikethrough line
  gContext.moveTo(startX, startY);
```

www.it-ebooks.info

```
  gContext.lineTo(endX, endY);
  gContext.closePath();
  // make strikethrough line permanent
  gContext.stroke();
}
```

Before drawing our strikethrough, we convert the `column` and `row` properties of the *`startCell`* argument (that is passed into the function) into x and y coordinates representing the center of the appropriate cell. We do the same for the *`endCell`* argument. As before, we set up the `lineWidth` and `strokeStyle` properties to match what we did in the SVG version. We then call the `beginPath` method to start the drawing of our strikethrough. We use a single `moveTo` call to position our pencil in the center of the appropriate starting cell, and then call `lineTo` to draw a line to the center of our ending cell. The `closePath` method is called to end the drawing of our strikethrough. We call the `stroke` method to make the strikethrough permanent.

Defining the SVG "Game Over" Text

In our SVG document, we'll need to define a `text` element (positioned in the center of the grid) to communicate when a player has won or the game has been drawn:

noughts_and_crosses.svg *(excerpt)*

```
<text id="gameOverText" x="150" y="150"
    font-family="Palatino, Palatino Linotype, Georgia,
➥Times New Roman, serif"
    font-size="30" font-weight="bold" text-anchor="middle"
    dominant-baseline="middle"></text>
```

Our `text` element is given an `id` of `gameOverText`, so that we can dynamically add the appropriate win/draw text to it using JavaScript at the end of the game. It's positioned in the center of the 300×300 playing grid by assigning 150 to both `x` and `y` attributes. The `font-family` attribute is a comma-separated list of fonts uses to style the text: if the first font is not present on the user's system, the next font in the list is used. It is wise to end the list with the name of a general font type (for example, serif or sans-serif) as a fallback if none of the previous fonts are present on the user's system. The text is sized and emboldened using the `font-size` and `font-weight` attributes. Setting the `text-anchor` and `dominant-baseline` attributes to `middle` ensures that the center of the actual text string corresponds to the `x` and `y` attributes

www.it-ebooks.info

of the `text` element, essentially aligning the text both horizontally and vertically. JavaScript will be used to assign a color by setting the `fill` attribute.

Although not essential to the core workings of the game, it would be nice to make the text more easily readable when it's overlaid on the grid. Therefore, we'll insert a semi-transparent background behind the text that will allow the grid to be seen, but will create more contrast with the text. SVG provides a very useful `linearGradient` element that we can use to create an eye-catching semi-transparent gradient background for our text:

noughts_and_crosses.svg *(excerpt)*

```
<!-- gradient for game over background -->
<linearGradient id="gradient1" x1="0" y1="0" x2="0" y2="100%">
  <stop id="stop1" offset="0" stop-color="white"
      stop-opacity="0.85" />
  <stop id="stop2" offset="0.5" stop-color="white"
      stop-opacity="0.95" />
  <stop id="stop3" offset="1" stop-color="white"
      stop-opacity="0.85" />
</linearGradient>
```

We use the `x1`, `y1`, `x2`, and `y2` attributes to define the direction in which the gradient should be drawn; in our case, we want to create a vertical gradient, which we do by defining a vertical line from 0,0 to 0,100%. We define three `stop` elements, each with an `id` attribute set so that we can manipulate the colors using JavaScript; it allows color (`stop-color`) and opacity (`stop-opacity`) to be set at specific points along the way (offset). Using offset values of 0, 0.5, and 1, we dictate the color and opacity at the beginning, middle, and end of our gradient. The SVG code above creates a white gradient that's more opaque in the middle and more transparent at the beginning and end (that is, the top and bottom of a vertical gradient).

We've defined a linear gradient, but we're yet to make use of it. So, we'll define a rectangle that covers the 300×300 playing grid, and make sure that it uses the gradient as its fill by referencing the gradient's `id`:

noughts_and_crosses.svg *(excerpt)*

```
<rect id="gameOverBackground" x="0" y="0" height="300" width="300"
    stroke="none" fill="url(#gradient1)" />
```

www.it-ebooks.info

Putting the semi-transparent gradient background and the "game over" text together, we can create a game over notice to display at the end of the game:

noughts_and_crosses.svg *(excerpt)*

```
<!-- game over notice -->
<g id="gameOverNotice">
  <rect id="gameOverBackground" x="0" y="0" height="300" width="300"
      stroke="none" fill="url(#gradient1)" />
  <text id="gameOverText" x="150" y="150"
      font-family="Palatino, Palatino Linotype, Georgia,
➥Times New Roman, serif"
      font-size="30" font-weight="bold" text-anchor="middle"
      dominant-baseline="middle"></text>
</g>
```

Drawing the Game Over Text

In our JavaScript file for the SVG example, we need to create a function we can call to set the "game over" text, draw the game over notice overlaying the playing grid, and draw the text on top of that—let's call it `drawGameOverNotice`. What this function should do is retrieve the game over text element, dynamically set the game over text depending on whether the game was won or drawn, insert the text into the text element, set the gradient colors of the game over notice, and create a new `use` element which references the game over notice group element we defined above. Let's first look at how we can set up the text:

noughts_and_crosses_svg.js *(excerpt)*

```
var COLOUR_GAME_OVER_TEXT = 'black';  // colour of game over text

function drawGameOverNotice(gameWon) {
  // text to display when game is over
  var gameOverText;
  // SVG text element into which game over text will be inserted
  var gameOverTextElement;
  // retrieve game over text element
  gameOverTextElement = gSvgDoc.getElementById('gameOverText');
  if (gameOverTextElement) {
    gameOverTextElement.setAttributeNS(null, 'fill',
➥COLOUR_GAME_OVER_TEXT);
    // set appropriate game over text
    switch (gameWon) {
```

www.it-ebooks.info

```
    case 1:
      gameOverText = 'Player 1 WINS!';
      break;
    case 2:
      gameOverText = 'Player 2 WINS!';
      break;
    default:
      gameOverText = 'DRAW!';
    }
    // replace entire contents of game over text element
    // with game over text
    while (gameOverTextElement.firstChild !== null) {
      // remove all existing content
      gameOverTextElement
          .removeChild(gameOverTextElement.firstChild);
    }
    gameOverTextElement
        .appendChild(gSvgDoc.createTextNode(gameOverText));
  }
  // draw game over notice
  ⋮
}
```

We use the `getElementById` function to retrieve the game over text element (stored in the `gameOverTextElement` variable). If the element has been successfully found in the SVG document, we ensure that the text color is set by assigning `COLOUR_GAME_OVER_TEXT` (black) to the `fill` attribute of the `text` element. We then set the text this element will contain (using the `gameOverText` variable) depending on the value of the *`gameWon`* argument passed into this function. The *`gameWon`* argument can contain one of three values:

- 0 indicates a drawn game
- 1 indicates that Player 1 has won the game as three noughts in a line have been found
- 2 indicates that Player 2 has won the game as three crosses in a line have been found

It's worth noting that to display text in our SVG document, we need to first create a text node (to store the characters of our text), and then insert that text node into a `text` element. Before this happens, we need to make sure that the `text` element

www.it-ebooks.info

is emptied of all content. We do this by looping through and removing any of the `text` element's children, retrieving each child using the `text` element's `firstChild` property and passing them one at a time into the `text` element's `removeChild` method. Finally, we create our text node (using the text we stored in our `gameOverText` variable) and append it to the empty text element (`gameOverTextElement`).

Now let's look at displaying a semi-transparent layer between the playing grid and the text element:

noughts_and_crosses_svg.js *(excerpt)*

```
// colour of game over text background gradient 1
var COLOUR_GAME_OVER_TEXT_BGD1 = 'rgb(240, 240, 240)';
// colour of game over text background gradient 2
var COLOUR_GAME_OVER_TEXT_BGD2 = 'rgb(200, 200, 200)';

function drawGameOverNotice(gameWon) {
  ⋮
  // draw game over notice
  // - first set gradient stop colours
  gSvgDoc.getElementById('stop1')
    .setAttributeNS(null, 'stop-color', COLOUR_GAME_OVER_TEXT_BGD1);
  gSvgDoc.getElementById('stop2')
    .setAttributeNS(null, 'stop-color', COLOUR_GAME_OVER_TEXT_BGD2);
  gSvgDoc.getElementById('stop3')
    .setAttributeNS(null, 'stop-color', COLOUR_GAME_OVER_TEXT_BGD1);
  // then use game over notice element which uses
  // above gradient stop colours
  newUseElement = gSvgDoc.createElementNS(SVG_NS, 'use');
  newUseElement
    .setAttributeNS(XLINK_NS, 'href', '#gameOverNotice');
  gGameContainer.appendChild(newUseElement);
}
```

We use the familiar `getElementById` method to retrieve the three `stop` elements that form our game over notice gradient. We then set the stop colors at the top and bottom of the gradient to a very light gray, and the stop color in the middle of the gradient to a slightly darker gray. We finally create a new SVG `use` element (which refers to the game over notice group element we defined earlier) and insert it into our `gameContainer` group element:

www.it-ebooks.info

```
<g id="gameContainer">
  <use stroke="silver" x="0" y="0" xlink:href="#grid" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none"
      cx="50" cy="50" />
  <use stroke="red" x="115" y="115" xlink:href="#cross" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none"
      cx="150" cy="50" />
  <use stroke="red" x="215" y="115" xlink:href="#cross" />
  <circle r="35" stroke="blue" stroke-width="7" fill="none"
      cx="250" cy="50" />
  <line x1="50" y1="50" x2="250" y2="50" stroke="orange"
      stroke-width="7" fill="none" />
  <use xlink:href="#gameOverNotice" />
</g>
```

This SVG code will overlay the "game over" notice on the playing grid, as in Figure 9.8.

Figure 9.8. A "game over" notice overlaying the playing grid in SVG

In our JavaScript file for the HTML5 canvas example, we also need a `drawGameOverNotice` function. What this function should do is produce the same output as the SVG version in Figure 9.8, but in a subtly different way. The setting of the "game over" text (based on the value of the *`gameWon`* argument) has been omitted, as it is identical to the SVG code:

www.it-ebooks.info

noughts_and_crosses_canvas.js *(excerpt)*

```
function drawGameOverNotice(gameWon) {
  // gradient fill to apply to game over background
  var gameOverGradient;
  // text to display when game is over
  var gameOverText;
  // width of text displayed when game is over
  var gameOverTextWidth;
  // set appropriate game over text
  ⋮
  // draw background box for text
  gameOverGradient = gContext.createLinearGradient(0, 0, 0,
➥gGridHeight);
  gameOverGradient.addColorStop(0, COLOUR_GAME_OVER_TEXT_BGD1);
  gameOverGradient.addColorStop(0.5, COLOUR_GAME_OVER_TEXT_BGD2);
  gameOverGradient.addColorStop(1, COLOUR_GAME_OVER_TEXT_BGD1);
  gContext.fillStyle = gameOverGradient;
  gContext.fillRect(0, 0, gGridWidth, gGridHeight);
  // draw text overlaying background box
  gameOverTextWidth = gContext.measureText(gameOverText).width;
  gContext.fillStyle = COLOUR_GAME_OVER_TEXT;
  gContext.fillText(gameOverText, (gGridWidth - gameOverTextWidth)
➥/ 2, gGridHeight / 2, gGridWidth);
}
```

We create our vertical linear gradient in a very similar way to our SVG version. We call the `createLinearGradient` method passing the coordinates of a vertical line from the top left of our canvas (0,0) to the bottom left of our canvas (0,`gGridHeight`) in order to specify the direction of the gradient. We add three color stops, as before, at the top, middle, and bottom of the vertical gradient. We set the `fillStyle` property to use the linear gradient we've created as the fill color of the next filled item. We then draw and fill a rectangle covering the playing grid with our gradient using the `fillRect` method. We gauge the horizontal space that our text will occupy, set our text color to black (the value stored in `COLOUR_GAME_OVER_TEXT`), and draw and fill the text in the center of our playing grid using the `fillText` method.

Well, that's all the building blocks we'll need to draw in our Noughts & Crosses game. Next up: the game logic.

www.it-ebooks.info

Developing the Game Logic

To pull all the building blocks together to form our Noughts & Crosses game, we'll need to develop a number of additional JavaScript functions to implement our game logic. We'll have to initialize the game at the outset to set up and configure our SVG or HTML5 canvas, so that it's ready for drawing on and able to respond to mouse clicks. We'll need to allow the following: a new game to be started, the canvas to be cleared, the mouse cursor position to be detected when a click occurs, an empty cell to be detected, three noughts or three crosses in a line to be recognized, and a game to be ended. The name and purpose of each function is listed in the table below.

www.it-ebooks.info

Table 9.3. Game logic functions

Function	Purpose (SVG version)	Purpose (HTML5 canvas version)
`initialiseGame`	■ retrieve the SVG document ■ assign an onclick event handler to the SVG document to respond to mouse clicks ■ set up the grid dimensions ■ set up the cell dimensions ■ retrieve the `gameContainer` group element, which will contain all dynamically created SVG elements ■ start a new game	■ retrieve the `canvas` element ■ assign an `onclick` event handler to the canvas element to respond to mouse clicks ■ set up the grid dimensions ■ set up the cell dimensions ■ initialize the two-dimensional drawing context for the `canvas` element ■ set up font properties for outputting the "game over" text to the `canvas` element ■ start a new game
`newGame`	■ clear the canvas ■ initialize the game variables ■ create a 3×3 array to record the locations of turns taken ■ draw the playing grid	[identical to SVG version]

www.it-ebooks.info

Function	Purpose (SVG version)	Purpose (HTML5 canvas version)
`canvasOnClick`	■ determine the cell in which the mouse cursor was clicked ■ check the cell is a valid one ■ check the cell is empty ■ draw a nought or cross in the cell as appropriate ■ update the game variables ■ determine whether the game has been one ■ end the game if the game has been won or the maximum number of turns have been taken	[identical to SVG version]
`clearCanvas`	■ remove all child elements of the `gameContainer` group element into which all dynamically created elements will have been inserted ■ insert an SVG `rect` element as the first element (to sit behind the playing grid) to ensure that Internet Explorer responds to mouse clicks in the grid cells	■ call the `clearRect` method

www.it-ebooks.info

Function	Purpose (SVG version)	Purpose (HTML5 canvas version)
`getCursorPosition`	■ determine the x and y coordinates of the mouse click in the SVG document ■ calculate which cell the coordinates correspond to (if within the bounds of the grid)	■ determine the x and y coordinates of the mouse click in the HTML document ■ determine the x and y coordinates relative to the canvas element (rather than the entire HTML document) ■ calculate which cell the coordinates correspond to (if within the bounds of the grid)
`isEmptyCell`	■ check whether the particular cell is empty	[identical to SVG version]
`threeInALine`	■ check whether the minimum number of turns to win a game (five turns) have been taken ■ check for three noughts or three crosses in a line ■ draw an appropriate strikethrough if three in a line were found	[identical to SVG version]
`endGame`	■ indicate that the game has ended ■ display the "game over" notice indicating the outcome of the game	[identical to SVG version]

We won't go through every line of code for all the functions listed above (the complete code for both SVG and HTML5 Canvas versions can be found online).[3] Instead, we'll take a look at the particularly interesting parts of each function.

[3] http://clive.wickhub.co.uk/books/community/

www.it-ebooks.info

The `initialiseGame` Function

Let's start with the SVG version of `initialiseGame`:

noughts_and_crosses_canvas.js *(excerpt)*

```
function initialiseGame(svgContainer) {
  if (svgContainer) {
    // get SVG document
    if (typeof svgContainer.getSVGDocument !== 'undefined') {
      gSvgDoc = svgContainer.getSVGDocument();
    } else {
      gSvgDoc = svgContainer.contentDocument;
    }
    // check document element is actually an svg element
    if (gSvgDoc.documentElement.nodeName === 'svg') {
      // add event handler to SVG document so as to
      // respond to user mouse clicks
      ⋮
      // setup grid dimensions
      ⋮
      // setup cell dimensions
      ⋮
      // get group element in which to draw everything
      ⋮
      // start a new game
      ⋮
    }
  }
}
```

The *`svgContainer`* argument passed into the function is an HTML `object` element, which references our external SVG document (**noughts_and_crosses.svg**). In order to gain direct access to the contents of this SVG document, we call the `getSVGDocument` method that "provides access to an SVG document embedded by reference in another DOM-based language."[4] However, `getSVGDocument` is deprecated (as of the second edition of the SVG 1.1 Specification—a working draft) in favor of the `contentDocument` property, so it's wise for us to ensure `getSVGDocument` is defined before using it and, as a fallback, try the `contentDocument` instead. The same specification also advises checking that the document element (that is, the root node) that is returned is an `svg` element.

[4] http://www.w3.org/TR/SVG/struct.html#InterfaceGetSVGDocument

www.it-ebooks.info

We use the following code to attach an event handler to respond to mouse clicks:

noughts_and_crosses_canvas.js *(excerpt)*

```
// add event handler to SVG document so as
// to respond to user mouse clicks
if (gSvgDoc.addEventListener) {
  // standard W3C method for assigning event handler
  gSvgDoc.addEventListener('click', canvasOnClick, false);
} else if (gSvgDoc.attachEvent) {
  // IE method for assigning event handler
  gSvgDoc.attachEvent('onclick', canvasOnClick);
}
```

We first check whether the standards-compliant way of attaching an event to an element is available to us: `addEventListener`. If it is, we use it to make sure our `canvasOnClick` function is called whenever the SVG document is clicked. Unfortunately, Internet Explorer doesn't support `addEventListener`, so we have to use IE's `attachEvent` method to make our `canvasOnClick` function the event handler for clicks in the SVG document. Note that we must include the `on` prefix when we pass the first argument to `attachEvent`, whereas `addEventListener` doesn't require it. Despite `attachEvent` being the correct method to utilize for IE, the combination of IE and Adobe SVG Viewer (ASV) seems to ignore it. So, we must improvise by creating an SVG `rect` element that sits behind the playing grid and has its `onclick` attribute set to `canvasOnClick` (this happens in our `clearCanvas` function—more on this later).

Next, we set up our grid and cell dimensions:

noughts_and_crosses_canvas.js *(excerpt)*

```
var PADDING_MULTIPLIER = 0.15;  // denotes 15% padding
⋮
// setup grid dimensions
if (svgContainer.width > svgContainer.height) {
  gGridWidth = svgContainer.height;
} else {
  gGridWidth = svgContainer.width;
}
gGridHeight = gGridWidth;
// setup cell dimensions
```

www.it-ebooks.info

```
gCellWidth = gGridWidth / 3;
gCellHeight = gGridHeight / 3;
gCellPadding = gCellWidth * PADDING_MULTIPLIER;
```

Our grid needs to be square, so we set the `gGridWidth` and `gGridHeight` variables equal to the same value. This value is the height or width of the element that contains our SVG document, depending on which is smaller. We then make our cell dimensions a third of the grid dimensions and calculate some cell padding as a percentage of the cell width, using a global variable called `PADDING_MULTIPLIER`.

Finally, we need to access our SVG group element (which has an id of `gameContainer`) where we'll insert all other SVG elements, then call the `newGame` function to prepare our SVG canvas for playing the game:

noughts_and_crosses_canvas.js *(excerpt)*

```
// get group element in which to draw everything
gGameContainer = gSvgDoc.getElementById('gameContainer');
// start a new game
newGame();

Our HTML5 Canvas version looks like this:
function initialiseGame(canvasElement) {
  // store reference to canvas element
  gCanvas = canvasElement;
  // ensure canvas element is in DOM and that browser supports it
  if (gCanvas && gCanvas.getContext) {
    // initialise 2-dimensional drawing context for canvas element
    gContext = gCanvas.getContext('2d');
    // add event handler to canvas element
    // to respond to user mouse clicks
    ⋮
    // setup grid dimensions
    ⋮
    // setup cell dimensions
    ⋮
    // setup font properties for outputting game over text
    // to canvas element at an appropriate size
    ⋮
    // start a new game
    ⋮
  }
}
```

www.it-ebooks.info

The *canvasElement* argument passed into the function provides access to the canvas element from which the two-dimensional context can be obtained, as described in an earlier section. Adding an event handler to our HTML5 canvas element uses the same technique as above (except that gSvgDoc is replaced by a variable called gCanvas, which represents the HTML5 canvas element). Setting up grid and cell dimensions is identical to the SVG version, as is starting a new game. The only additional part of this function is to set up our font properties for later outputting our game over text. In SVG, you may remember we set up our font when we defined our text element, so we need to do the equivalent for our HTML5 canvas version:

noughts_and_crosses_canvas.js *(excerpt)*

```
// denotes font size for text output, relative to grid size
var FONT_SIZE_MULTIPLIER = 0.1;
⋮
// setup font properties for outputting game over text to canvas
// element at an appropriate size
gFontSize = FONT_SIZE_MULTIPLIER * gGridWidth;
gContext.font = 'bold ' + gFontSize + 'px Palatino,
➥"Palatino Linotype", Georgia, "Times New Roman", serif';
gContext.textBaseline = 'middle';
```

We set the font size relative to our grid width, so that text will always be displayed at an appropriate size. We then assign font style, font size, and font family to the font property of the canvas context variable. Finally, we set the baseline of our text to middle, which essentially ensures the vertical center of our text sits on the y coordinate that we'll later use to position the text.

The newGame Function

Our initialiseGame function referred to two functions, newGame and canvasOnClick, which we'll look at next. Both functions are identical in SVG and HTML5 canvas versions of our game. First up, is the newGame function:

```
var EMPTY_CELL_MARKER = '-';    // denotes an empty cell

function newGame() {
  clearCanvas();
  // initialise game variables
  gGameOver = false;
  gNoughtsTurn = true;
```

www.it-ebooks.info

```
  gNumTurns = 0;
  // create array which will record locations of turns taken
  gTurns = [
    [EMPTY_CELL_MARKER, EMPTY_CELL_MARKER, EMPTY_CELL_MARKER],
    [EMPTY_CELL_MARKER, EMPTY_CELL_MARKER, EMPTY_CELL_MARKER],
    [EMPTY_CELL_MARKER, EMPTY_CELL_MARKER, EMPTY_CELL_MARKER]
  ];
  // draw playing grid
  drawGrid();
}
```

The `newGame` function first clears the canvas by unsurprisingly calling the `clearCanvas` function. We initialize our game variables ensuring that: the game over flag (`gGameOver`) is `false` at the start of a new game, the noughts player has first turn (`gNoughtsTurn` equals `true`), and the total number of turns recorded (`gNumTurns`) is reset to zero. We create a 3×3 array that we'll use throughout each game to record where the noughts and crosses have actually been drawn. We fill each position in the array with the value stored in `EMPTY_CELL_MARKER`. Finally, we make a call to draw our playing grid.

The `canvasOnClick` Function

The `canvasOnClick` function is effectively our game controller and looks like this:

```
var MAX_TURNS = 9;  // denotes maximum number of turns allowed

function canvasOnClick(evt) {
  var cell, gameWon;
  // determine cell in which mouse cursor was clicked
  cell = getCursorPosition(evt);
  // if valid cell then draw nought or cross as appropriate
  if ((cell) && (!gGameOver)) {
    // first make sure cell is empty
    if (isEmptyCell(cell)) {
      if (gNoughtsTurn) {
        drawNought(cell);
      } else {
        drawCross(cell);
      }
      // switch flag ready for the next go
      gNoughtsTurn = !gNoughtsTurn;
      // increment turn count to indicate that turn has been taken
```

www.it-ebooks.info

```
      gNumTurns = gNumTurns + 1;
      // check grid to see if there's 3 pieces in a line
      gameWon = threeInALine();
      // if game is won, or all turns have been taken
      if ((gameWon) || (gNumTurns >= MAX_TURNS)) {
        endGame(gameWon);
      }
    }
  }
}
```

We first call `getCursorPosition` so that we can determine which cell corresponds to the x and y coordinates of the user's mouse click on our canvas. We pass it the event object (evt) that was passed to the `canvasOnClick` function, as this object allows us to access the x and y coordinates of the click. The `getCursorPosition` function returns a Cell object (comprising column and row properties) that we store in the cell variable. If a valid cell is returned and the game isn't over yet (that is, the `gGameOver` variable equals `false`), we make a call to `isEmptyCell` passing our cell variable to ascertain whether the cell can have a nought or cross drawn in it. If it's the turn of the noughts (`gNoughtsTurn` equals `true`), we call our `drawNought` function; otherwise, we call `drawCross`, and the cell is filled with the appropriate piece. We set `gNoughtsTurn` to its opposite (`true` becomes `false` or `false` becomes `true`) to ensure alternate turns are taken by the players. We also increment the total number of turns taken. We then call `threeInALine` to check whether either player has won after taking their turn, storing the result in the `gameWon` variable. Finally, if the game is won, or the number of turns has exceeded the maximum number of turns (which is nine on a 3×3 grid), `endGame` is called to finish the game.

The `clearCanvas` Function

Let's now take a look at the `clearCanvas` function called within `newGame`. Different approaches are required for each version of our game, so let's examine the SVG version:

noughts_and_crosses_svg.js *(excerpt)*

```
function clearCanvas() {
  // remove all child elements of the gameContainer group element
  while (gGameContainer.firstChild) {
    gGameContainer.removeChild(gGameContainer.firstChild);
```

www.it-ebooks.info

```
  }
  // the following rect must be inserted first in order for
  // IE to respond to click events
  if ((!gSvgDoc.addEventListener) && (!gSvgDoc.attachEvent)) {
    var backgroundElement = gSvgDoc.createElementNS(SVG_NS, 'rect');
    backgroundElement.setAttributeNS(null, 'height', gGridHeight);
    backgroundElement.setAttributeNS(null, 'width', gGridWidth);
    backgroundElement.setAttributeNS(null, 'x', 0);
    backgroundElement.setAttributeNS(null, 'y', 0);
    backgroundElement.setAttributeNS(null, 'fill', 'white');
    backgroundElement.setAttributeNS(null, 'onclick',
➥'canvasOnClick(evt)');
    gGameContainer.appendChild(backgroundElement);
  }
}
```

The `clearCanvas` function first needs to remove all SVG elements that have been dynamically inserted into the `gGameContainer` group element, so a `while` loop is used to remove the first child element within the group element until there are no more child elements. We noted earlier that IE doesn't behave as it should in response to mouse clicks. So, if it hasn't been possible to correctly attach an event handler to our SVG document, we create a brand new SVG `rect` element, position it in the top-left corner, and give it the same dimensions as our playing grid. We give it a white fill color, set its `onclick` attribute to refer to `canvasOnClick`, and insert the `rect` into `gGameContainer`. Any elements inserted into `gGameContainer` will be drawn on top of this `rect`. This workaround appears to address IE's deficiency.

Thankfully, our HTML5 canvas version is a simple one-line call to the `clearRect` method of the drawing context. We pass in 0,0 (the top left) as the starting x and y coordinates, and `gGridWidth` and `gGridHeight` as the width and height of the rectangle to be cleared—it's as easy as that:

noughts_and_crosses_canvas.js *(excerpt)*

```
function clearCanvas() {
  gContext.clearRect(0, 0, gGridWidth, gGridHeight);
}
```

www.it-ebooks.info

The `getCursorPosition` Function

An important function that we won't delve into here is `getCursorPosition`, which retrieves the x and y coordinates of the user's mouse click from the event object passed into it. It then checks that the click was within the bounds of the playing grid and, if so, uses some fairly straightforward math to determine the column and row of the corresponding cell. A new `Cell` object is created from this column and row information, and is stored in a local cell variable. If the click was "out of bounds," the `cell` variable is set to `null` (indicating an invalid cell when we check it in our `canvasOnClick` function). The function then returns the value stored in the `cell` variable. This function is implemented in a very similar way by both versions of our game.

The `isEmptyCell` Function

Another simple function, which is identical in both versions of our game, is the `isEmptyCell` function:

```
function isEmptyCell(cell) {
  // check whether the particular cell is empty and return result
  return gTurns[cell.column][cell.row] === EMPTY_CELL_MARKER;
}
```

The function is passed a Cell object, where the column and row properties are used to look up a particular item in our `gTurns` array. If the cell hasn't had anything drawn in it, it will still contain an `EMPTY_CELL_MARKER` value, so a check is made against this and a true or false value is returned by the function, as appropriate.

The `threeInALine` Function

The penultimate function we'll examine is `threeInALine`, which is used to check if the game has been won by either player after each has taken their turn. It's been simplified to clearly show the logic rather than the full detail of the function:

```
function threeInALine() {
  var threeInALineFound = 0, startCell, endCell;
  // check if enough turns taken for 3 pieces in a line to
  // be possible via alternate turns (minimum is 5 turns)
  if (gNumTurns > 4) {
    // loop through the 3 possible variations in both
```

www.it-ebooks.info

```
    // horizontal and vertical planes
    for (var i = 0; i < gTurns.length; i++) {
      // check for 3 pieces in a horizontal line (i.e. a row)
      ⋮
      // check for 3 pieces in a vertical line (i.e. a column)
      ⋮
    }
    // if 3 pieces in a horizontal/vertical line have NOT been found
    ⋮
    // check for 3 pieces in a diag. line (top-left to bottom-right)
    ⋮
    // check for 3 pieces in a diag. line (top-right to bottom-left)
    ⋮
    // if three in a line found then draw
    // strikethrough across appropriate cells
    ⋮
  }
  return threeInALineFound;
}
```

We first declare a variable called `threeInALineFound`, which can be given one of three possible values within our function:

- 0 indicates that no three adjacent cells contain noughts or crosses
- 1 indicates that three adjacent cells contain noughts
- 2 indicates that three adjacent cells contain crosses

Two further variables are also declared, `startCell` and `endCell`, which store the column and row of the first cell and the last cell when three adjacent cells contain noughts or crosses—either horizontally, vertically, or diagonally. These variables are used to draw a line (a strikethrough) through the three adjacent cells, so we need to know where to start drawing our line and where to finish it. Next, we check that at least five turns have been taken in total, as no game of noughts and crosses on a 3×3 grid can be won via alternate turns with four or less turns. This saves us going through our playing grid checking for three adjacent identical cells prematurely. If enough turns have been taken for a win to be possible, we loop through the three possible variations in both horizontal (top row, middle row, bottom row) and vertical (left column, middle column, and right column) planes, checking for three identical pieces. If three adjacent identical pieces are not found, we make a separate

www.it-ebooks.info

check for three diagonally adjacent identical pieces from top left to bottom right, then top right to bottom left. This covers all the potential three-in-a-line winning positions. Here's an example of checking for diagonally identical pieces in one direction:

```
// check for 3 pieces in a diagonal line (top-left to bottom-right)
if (((gTurns[0][0] === gTurns[1][1])
    && (gTurns[1][1] === gTurns[2][2]))
    && (gTurns[2][2] !== EMPTY_CELL_MARKER)) {
  startCell = new Cell(0, 0);
  endCell = new Cell(2, 2);
}
```

The above code checks that:

- the top-left cell is equal to the center cell
- the center cell is equal to the bottom-right cell
- the bottom-right cell is not empty (to make sure that the three identical cells are not all empty)

If these conditions are met, a new `Cell` object is created with a column of 0 and a row of 0, and is stored in the `startCell` variable. A second new `Cell` object is created with a `column` of 2 and a `row` of 2, and is stored in the `endCell` variable. These variables will be used to draw a line (a strikethrough) from the top-left cell of the grid to the bottom-right cell of the grid, thus highlighting where three pieces in a line were found:

```
// if three in a line found then draw strikethrough
// across appropriate cells

// check if startCell has been assigned
if (typeof startCell !== 'undefined') {
  // three noughts in a line
  if (gTurns[startCell.column][startCell.row] === 'O') {
    threeInALineFound = 1;
  // three crosses in a line
  } else if (gTurns[startCell.column][startCell.row] === 'X') {
    threeInALineFound = 2;
```

www.it-ebooks.info

```
  }
  drawStrikeThrough(startCell, endCell);
}
```

In the above code, the `startCell` variable is used to look up an item in our `gTurns` array of turns taken, to determine whether three adjacent noughts or three adjacent crosses were found, and our `threeInALineFound` variable is set accordingly. A call to `drawStrikeThrough` is then made to draw a line from the cell indicated by `startCell` to that indicated by `endCell`. The function then returns the `threeInALineFound` variable as the result of the function.

The endGame Function

The final function we'll look at is our `endGame` function:

```
function endGame(gameWon) {
  // indicate that the game has ended
  gGameOver = true;
  // display notice indicating outcome of game
  drawGameOverNotice(gameWon);
}
```

This function simply indicates that the game is over, and displays an appropriate notice indicating the result of the game by calling the `drawGameOverNotice` function.

Summary

There you have it! Two fully working versions of the game of Noughts & Crosses using SVG and, as an alternative, HTML5 canvas, and lashings of JavaScript.

In this tutorial, we've briefly looked at vector graphics in general, and SVG and HTML5 canvas in particular. We've stepped through the process of building a simple game of Noughts & Crosses in both technologies. This tutorial has hopefully provided a spark to ignite your interest in SVG and canvas, as we've examined some of the basics of each technology. Remember that implementations of both technologies in the major web browsers will continue to progress and advance, so be sure to keep testing everything you create for cross-browser compatibility, and be ready to revisit workarounds for nonstandard browser behavior as browsers move closer to standardization.

www.it-ebooks.info

There are plenty of ways to take this game forward (or even use it as a basis for an entirely novel application). This tutorial, I hope, has whet your appetite for learning more, so I've provided some ideas for taking things further in the next section.

I've also listed some useful resources on the Web—primers, tutorials, official standards, articles, and compatibility tables—for both SVG and canvas, and a number of invaluable tools for working with SVG in particular.

Live versions of each game can be played at http://clive.wickhub.co.uk/books/community/, where you can also download the complete code.

Taking It Further

There are a host of ways in which this Noughts & Crosses game can be extended, developed, and improved, some of which I've suggested below.

In this tutorial, we've had a look at using simple linear gradients. The aesthetics of this game could be enhanced through gradients by incorporating more stop colors, varying the angles of the gradients, using radial gradients (which radiate out from a center point) instead of linear ones, or applying gradients to some of the other building blocks of the game. Patterns, rather than solid fill colors, can also be used to liven up parts of the game. SVG allows masks and filters to be applied to elements, which could possibly be used to good effect in the game.

SVG offers support for quickly and easily animating elements using the Synchronized Multimedia Integration Language (SMIL—pronounced "smile"), so noughts and crosses could slide in from different angles and speeds into appropriate cells, for example. Animation can also be achieved in canvas using JavaScript but, when compared with employing SMIL in SVG, this often requires writing many more lines of code to achieve the same animated effect.

The canvas version of our game sizes itself appropriately to a canvas element of any dimensions. The SVG version, however, will currently use a fixed 300×300 area, regardless of the size of the HTML object element which houses it. This is not ideal, but can be rectified by setting the width and height attributes of the `svg` element in our SVG document to 100%, and also applying the `viewBox` and `preserveAspectRatio` attributes to the `svg` element. The `viewBox` attribute allows us to separate the physical dimensions of our graphic (the space it takes up in our

www.it-ebooks.info

HTML document) from the internal coordinate space (the units used to position elements in our SVG document). Let's look at a quick example:

```
<svg height="100%" width="100%" viewBox="0 0 300 300"
    preserveAspectRatio="xMinYMin">
```

The `viewBox` attribute allows a rectangle to be specified using four space-separated values: x coordinate, y coordinate, width, and height. In our example, we've defined a 300×300 rectangle positioned in the top-left corner. This rectangle is then mapped to the `svg` element's `width` and `height` attributes (which in our case are set to 100% of the dimensions of the element in which our SVG document is embedded). Setting the `preserveAspectRatio` attribute to `xMinYMin` ensures that everything is scaled uniformly from the top-left corner. What this means in simple terms is that we can position all our elements within our SVG document as if the SVG canvas were 300×300, and our SVG canvas will be magically scaled up or down to fit the physical space it's given when embedded in an HTML document. For example, if the HTML object element that referenced our SVG document was 600×600 in size, the dimensions of our SVG document would be doubled so that the SVG canvas fitted the area allowed by the object element. This will unfortunately present us with difficulties in our `getCursorPosition` function, as we'd need to take account of this scaling. However, some scripts are available to assist us with the calculations we must use—for example, Kevin Lindsey's ViewBox script.[5]

An all-in-one approach could be taken, whereby the two versions of our JavaScript code could be merged into a single script that could draw onto an SVG or HTML5 canvas, depending on which of the two was passed to it. Alternatively, it could generate an SVG or HTML5 canvas depending on which of the technologies was supported by the particular browser running the script.

One glaringly obvious enhancement to the game would be to implement some kind of computer intelligence so that a single human player could play against the computer. Perhaps the human player could even select the difficulty level of their computer opponent.

[5] http://www.kevlindev.com/gui/utilities/viewbox/index.htm

www.it-ebooks.info

That's just a few ways to extend and improve our Noughts & Crosses game, but don't feel you have to limit yourself to these. Use your imagination and creativity, and who knows what you might produce.

Resources

An SVG Primer for Today's Browsers [David Dailey], http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

Excellent and comprehensive introduction to all things SVG.

Dive Into HTML5: Let's Call It a Draw(ing Surface) [Mark Pilgrim], http://diveintohtml5.org/canvas.html

Fantastic and easy-to-read introduction to the HTML5 Canvas element.

HTML Living Standard—The `canvas` element [WHATWG], http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html

The essential reference manual for the HTML5 `canvas` element—a working draft.

Scalable Vector Graphics (SVG) 1.1 (Second Edition) Working Draft [W3C], http://www.w3.org/TR/SVG11/.

The essential reference manual for SVG, a working draft.

SVG: Evolution, Not Revolution [Jeff Schiller], http://dev.opera.com/articles/view/svg-evolution-not-revolution/

Series of articles exploring how to integrate SVG into web applications.

SVG or Canvas? Choosing between the two [Mihai Sucan], http://dev.opera.com/articles/view/svg-or-canvas-choosing-between-the-two/.

Article exploring the differences between SVG and HTML5 Canvas, and the advantages and disadvantages of each.

SVG Support [Jeff Schiller], http://www.codedread.com/svg-support.php

Table of results of running SVG implementations in web browsers and browser plugins through the official SVG Test Suite.

When can I use …, http://caniuse.com/

Compatibility tables for features in HTML5, CSS3, SVG, and other upcoming web technologies.

www.it-ebooks.info

SVG Tools

Adobe Illustrator, http://www.adobe.com/products/illustrator/

A vector graphics editor that allows artwork to be imported and exported in SVG format.

Inkscape, http://inkscape.org/

An open source vector graphics editor that uses SVG as its native file format, and offers an alternative to Adobe Illustrator.

Amaya, http://www.w3.org/Amaya/

A web browsing and authoring environment with support for editing and viewing SVG.

Apache Batik, http://xmlgraphics.apache.org/batik/

A toolkit for manipulating SVG images in Java applications.

Google SVG Web, http://code.google.com/p/svgweb/

A JavaScript library that provides SVG support on many browsers.

jQuery SVG, http://keith-wood.name/svgRef.html

A jQuery plugin that allows interaction with an SVG canvas.

Raphael, http://raphaeljs.com/

A small JavaScript library that simplifies working with vector graphics on the Web.

www.it-ebooks.info

www.it-ebooks.info

Chapter 10

Efficient ActionScript

by Christian Snodgrass

The number of tips, tricks, and techniques to get the most out of ActionScript could (and have) filled many books. Here is a selection of some of the more useful approaches for use in mid-to-large projects.

Object-oriented ActionScript

If you aren't familiar with object-oriented programming, or are yet to try object-oriented programming in ActionScript 3, you've been missing out. ActionScript 3 introduced object-oriented programming to ActionScript, which helped boost it into a *real* programming language.

All the concepts of object-oriented programming could fill a book (or several), but the main concept to understand is class. A **class** basically arranges all functions and variables into logical groups, which are then used to create objects. `MovieClip` and `Sprite` are two common built-in ActionScript 3 classes. If you're unfamiliar with classes, a number of online resources and books cover them in detail.

www.it-ebooks.info

Screens

When you're creating your entire program in one frame (or with the Flex compiler), you don't have multiple frames to organize your various objects. To make it easy to keep track of your objects, it's a good idea to create different screen classes for each screen.

A screen is a simple class, it just has to extend `Sprite`. Then, in the class, you can add all your images, text fields, buttons, and anything else you need to create your screen.

Since the screens are self-contained, it becomes a simple task of adding and removing the screens when you want to change them.

Use a Single Enter Frame Event Listener

In ActionScript 3, you frequently use an enter frame event listener for game loops and other similar tasks.

It is possible to add any number of event listeners to different objects. At first, this may seem like a good idea because each object would be doing its own thing. However, more often then not, this can lead to trouble because you aren't always sure which will fire first, and matters could get out of sync. Additionally, it can make it much more difficult to debug problems that are happening within the handler functions.

An easier to manage and more reliable technique is to have only one enter frame event listener and handler. From this centralized location, you can control everything within your program.

Some of the benefits to this technique include:

- One location to manage and order all your enter frame code (though you can and should break the code into separate functions)
- If you wanted to stop the listener (for example, pausing a game), you can simply remove the event listener, or add a single `if` statement to check if it's paused
- You know exactly what is going to fire, and in what order.

www.it-ebooks.info

If you are also using the screen method described above, instead of giving each screen its own enter frame event listener, you can just give it a function (like `Update`), which is called on the active screen.

A simple way to do this would be to create a class that extends `Sprite` and implements the function. Each screen could then extend your class and override the `Update` function:

```
package {
  import flash.display.Sprite;

  public class MyScreen extends Sprite {
    public function MyScreen() {}
    public function Update() {}
  }
}
```

Singleton Pattern

The singleton pattern is not unique to Flash, but it's quite useful, nonetheless. A **singleton** is basically a class that can have one—and only one—instance of itself throughout the entire program. This allows you to create handy classes that can basically be used from anywhere in your program, using only the memory required for one of these classes.

The singleton pattern involves a static instance of a class, contained within the class, and a constructor that can only be called from within the class. The static instance is the singleton, the only instance of the class. Since the constructor can only be called within the class, it prevents new instances of the class from being created.

There are several approaches to creating singletons in ActionScript 3. The traditional method involves creating a protected constructor, so it can't be used outside the class; however, ActionScript 3 doesn't allow this, so a lock is often used. The lock is simply an internal class (a class which can't be used outside of that file).

The lock is then used as a required parameter of the constructor. Since the lock class can't be used outside the class, there's no way to call the constructor unless you pass `null` for the parameter. In your constructor, you can check if the lock is `null`; if it is, you throw an error to prevent this workaround.

www.it-ebooks.info

Since Flash also allows for getter functions, you can use these to automatically create the instance the first time it's needed. If the constructor requires a parameter to be passed from the outside (such as the stage), you could also create an initialize function that accepts these parameters and passes them along.

Here is a basic singleton template for classes that have no requirement for extra parameters:

```
package {
  public class MyClass {
    // Static instance of our class.
    protected static var _instance:MyClass = null;

    // Public getter for the instance.
    public static function get Instance():MyClass {
      // If the instance hasn't been created yet, go ahead
      //  and create it.
      if(_instance == null)
        _instance = new MyClass(new MyClassLock());

      // Return the instance.
      return _instance;
    }

    // Constructor for our class.
    public function MyClass(lock:MyClassLock) {
      // If the lock is null, throw an error because it's
      //  being called outside of the class.
      if(lock == null)
        throw new Error("You can't create this class directly.");

      // The rest of constructor code goes here.
    }

    // The rest of class code goes wherever you want, just
    //  like normal.
  }
}

// The lock class.
internal class MyClassLock {}
```

To use this singleton, you would just call `MyClass.Instance` for an instance of the class. You could then store this in a variable. From there, everything works just like

www.it-ebooks.info

using any other class. Just keep in mind that all the data is shared throughout the entire program.

If you need your class to take parameters, you could add an `Initialize` function, which needs to be called the first time you use the singleton, like so:

```
package {
  public class MyClass {
    // Static instance of our class.
    protected static var _instance:MyClass = null;

    // Public getter for the instance.
    public static function get Instance():MyClass {
      // If the instance hasn't been created yet, go ahead
      // and create it.
      if(_instance == null)
        throw new Error("Not initialized. Use Initialize() first.");

      // Return the instance.
      return _instance;
    }

    // Public static Initialize function which accepts the
    // parameters we need.
    public static function Initialize(param1:int):MyClass {
      // If the instance hasn't been created yet, go ahead
      // and create it.
      if(_instance == null)
        _instance = new MyClass(new MyClassLock(), param1);

      // Return the instance.
      return _instance;
    }

    // Constructor for our class.
    public function MyClass(lock:MyClassLock, param1:int) {
      // If the lock is null, throw an error because it's
      // being called outside
      // of the class.
      if(lock == null)
        throw new Error("You can't create this class directly.");

      // The rest of constructor code goes here.
    }
```

www.it-ebooks.info

```
    // The rest of class code goes wherever you want, just
    // like normal.
  }
}

// The lock class.
internal class MyClassLock {}
```

The only difference from the previous version is that you have to call `MyClass.Initialize()` before you can use `MyClass.Instance`; however, for all subsequent times, you can just use `MyClass.Instance`. If you try to use `Instance` first, it throws an error.

Remember when you create your own singleton to make sure that you change all the class references to match the class you're creating.

Unfortunately, you can't create a distinct singleton class that can be extended by any class you want to make a singleton, because that would require generic (templated) classes, which ActionScript 3 doesn't support.

Be sure to carefully consider whether you need to make a class a singleton or not. It can be very easy to get carried away. Once a singleton is created, it stays in memory for the duration of the program (unless you destroy it by setting all references to the instance to `null`), so too many singletons can use a lot of memory.

Globalized Stage

In ActionScript 3, the main class has a variable called `Stage`. This variable gives you information about the stage that have many uses. One such use is to obtain the width and height of the stage to position your other objects.

Normally, you would have to pass the stage all over the place to be able to use these variables. Fortunately, it's quite simple to create a singleton class that keeps track of this instance, thus allowing you to use it anywhere.

Such a class may look like this:

www.it-ebooks.info

```
package {
  import flash.display.Stage;

  public class StageManager {
    protected static var _instance:StageManager = null;
    public static function get Instance():StageManager {
      if(_instance == null)
        throw new Error("Not initialized. Use Initialize() first.");
      return _instance;
    }

    public static function Initialize(stage:Stage):StageManager {
      if(_instance == null)
        _instance = new StageManager(new StageManagerLock(), stage);
      return _instance;
    }

    public static function get TheStage():Stage {
      return Instance.TheStage;
    }

    protected var _stage:Stage = null;

    public function StageManager(lock:StageManagerLock,
➥stage:Stage) {
      if(lock == null)
        throw new Error("You can't initialize this
➥class directly.");
      _stage = stage;
    }

    public function get TheStage():Stage { return _stage; }
  }
}

internal class StageManagerLock {}
```

This class simply takes in the stage through the `Initialize` function, which would be called from the main class. Then, whenever you want the stage, you just need to call `StageManager.TheStage`. Now you don't have to worry about passing the stage throughout the code.

www.it-ebooks.info

Centralized Keyboard Input

Perhaps one of the most aggravating aspects about ActionScript 3, to both beginners and pros, is trying to manage keyboard input properly. One way to alleviate this hassle is to centralize your keyboard input into one place.

A great technique for doing this is to create a singleton class that has an object that always maintains the stage's focus (through the use of a focus out event listener), as well as an array that keeps track of which keys are down and up.

You can also implement `IsKeyDown` and `IsKeyUp` functions, which, as they suggest, allow you to check if a specified key is down or up.

The class may look a little like this:

```
package {
  import flash.display.Stage;
  import flash.events.FocusEvent;
  import flash.events.KeyboardEvent;

  public class InputManager {
    // Basic singleton stuff.
    protected static var _instance:InputManager = null;
    public static function get Instance():InputManager {
      if(_instance == null)
        _instance = new InputManager(new InputManagerLock());
      return _instance;
    }

    // Holds the key codes of all down keys.
    protected var _downKeys:Vector.<uint>;

    // The sprite which we will focus all keyboard input on.
    protected var _input:Sprite;

    public function InputManager(lock:InputManagerLock) {
      if(lock == null)
        throw new Error("You can't create this class directly.");

      // Initialize our variables.
      _downKeys = new Vector.<uint>();
      _iInput = new Sprite();
```

www.it-ebooks.info

```
      // If you don't use the StageManager singleton, pass
      // the stage as a variable.
      var stage:Stage = StageManager.TheStage;

      // Set the stage's focus to be our focus sprite.
      stage.focus = _input;

      // Add a FOCUS_OUT, KEY_DOWN, and KEY_UP event
      // listener.
      _input.addEventListener(
        FocusEvent.FOCUS_OUT,
➥focusOutHandler, false, 0, true);
      _input.addEventListener(KeyboardEvent.KEY_DOWN,
➥keyDownHandler, false, 0, true);
      _input.addEventListener(KeyboardEvent.KEY_UP,
➥keyUpHandler, false, 0, true);
    }

    // Returns true if the specified key is pressed down (it's
    // in our vector).
    public function IsKeyDown(keyCode:uint):Boolean {
      return (_downKeys.indexOf(keyCode) != -1);
    }

    // Returns true if the specified key is not pressed down
    // (it's not in our vector).
    public function IsKeyUp(keyCode:uint):Boolean {
      return (_downKeys.indexOf(keyCode) == -1);
    }

    // If _input loses the focus, it gets it back.
    protected function focusOutHandler(e:FocusEvent):void {
      stage.focus = _input;
    }

    // When a key is pressed, if it isn't in our vector
    //  already, we add (push) it.
    protected function keyDownHandler(e:KeyboardEvent):void {
      if(_downKeys.indexOf(e.keyCode) == -1)
        _downKeys.push(e.keyCode);
    }

    // When a key is released, if it's in our vector, we
    //  remove (splice) it.
    protected function keyUpHandler(e:KeyboardEvent):void {
```

www.it-ebooks.info

```
      var index:int = _downKeys.indexOf(e.keyCode);

      if(index == -1)
        return;
      _downKeys.splice(index, 1);
    }
  }
}
```

This approach allows you not only to check key input from anywhere without having to worry about what has focus, but also makes it easy to check if multiple keys are down (for example, you can check if the left and up arrow keys are pressed at the same time).

It becomes as simple as getting the instance, and then using `IsKeyUp` and `IsKeyDown` in your game loop when you're interested in what keys are pressed. This approach may be a bit odd to ActionScript 3-only developers that are accustomed to using listeners, but it will be familiar to those who have programmed in other languages, including ActionScript 2.

There is one minor drawback: elements that might normally require focus (such as text inputs) will always lose focus. To overcome this, you can simple add focus out and focus in events to your text input; this could stop the `InputManager` from re-claiming focus (when the text input is focused in) and then let it reclaim focus when you're done (when the text input focus outs).

Conclusion

In many ways, ActionScript 3 is like any other language out there—and in many ways, it's not. This means that many tried-and-true program patterns work great in ActionScript. At the same time, it indicates that there are many design patterns that need to be tweaked to work with ActionScript's special structure.

The structure of ActionScript makes it easy to stick bits of code here, there, and everywhere, so it's important to rein in this potential chaos. Centralizing many aspects of your work—such as main loop and keyboard input, making valuable classes like `Stage` globally available, and organizing your code into self-contained stages—are but a few ways to bring order to your code.

www.it-ebooks.info

ActionScript 3 has developed into an impressive language for many applications, be they web, desktop, or phone. It's important every once in a while to stop and consider what's going on behind the scenes, and seek out ways to work smarter for a more productive ActionScript experience.

www.it-ebooks.info

www.it-ebooks.info

Chapter 11

Databases: The Basic Concepts

by Nuria Zuazo

Once upon a time, a few geeks wanted to learn about databases while the rest of the population lived quite happily in ignorance. Today, databases gain popularity by the day; even people with no computer expertise realize how important they are to a business. It's understandable. With the growth of the Internet and its services, anyone can have their own website and online store, and databases are a must for any dynamic site. As a result, some basic knowledge is necessary to make the most of your database.

So, what exactly is a database?

A database is a collection of organized information, so any collection of organized information is a database. A phone book is a database. Surprised? Well, not all databases are computer-based, though, that's the type we're going to discuss here. After all, we spend most of our time and effort in front of a computer dedicated to the Web.

There are many types of databases and ways to classify them. For now, we'll concentrate on one particular type: relational databases. Relational databases are widely

www.it-ebooks.info

used and have many advantages. The software to handle and manipulate relational databases is called Relational Database Management System or RDBMS.

Relational Databases

The popularity of relational databases is due to the way they work to match data, making it much easier to find specific information.

Before relational databases existed, information used to be held in a text file, with each entry separated from the next by a character, for example:

```
Customer_ID, Name, LastName, City, Country, Age|1, Jamie, Smith,
Houston, USA, 24|2, Susan, Taylor, London, UK, 35|3, Marta, Lopez,
Madrid, Spain, 41|4, Freddie, Schmidt, Berlin, Germany, 28
```

As you can see, the information is hard to read, so searches could take quite a bit of time.

In a relational database, the information is kept in tables where the columns represent fields and the rows are records, as seen in Figure 11.1. Records are lists of related information—after all, in a relational database, relations are everything.

TABLE 1: CUSTOMERS

Customer_ID	Name	LastName	City	Country	Age
1	Jamie	Smith	Houston	USA	24
2	Susan	Taylor	London	UK	35
3	Marta	Lopez	Madrid	Spain	41
4	Freddie	Schmidt	Berlin	Germany	28

Figure 11.1. A table in a relational database

Tables provided a better way to present information, and were much easier to read. But there were greater gains to be had.

As I said, all the information in a record, by definition, is related, but relational databases can have another level of relationship. A table could have a relationship with another table. This meant that it was possible to have multiple tables and create relationships between them, making the database much more than a flat list of sequential information.

www.it-ebooks.info

So, how do we do this? Take the table above. With only four records, it's hardly a large table. Yet, it can grow, and new names will be added. It's logical, then, that two, three, or even thousands of new customers will live in the same country. Typing in the same country over and over could be a bit cumbersome. But there's no need with relational databases.

Instead, we can create a new table for the countries, as shown in Figure 11.2.

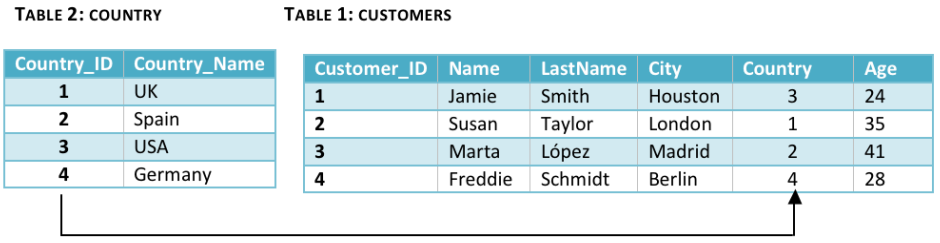

Figure 11.2. A relationship between two tables

By creating a relation between these two tables, I only have to write the number that represents that country—it's a real time saver. Relational databases provide a way that's easy to match information from different tables, as long as I'm able to build some kind of connection between them. It's a language that allows users to extract information from the database, as well as perform other tasks such as creating and deleting tables, and more. It's name is Structured Query Language, better known as SQL.

The relationship between tables is created by using two fields—one from each table—with the same attributes or properties, such as data type and field length. The data has to match somehow. In our example, it's easy to see that the number of the country is common in both tables, even if the name of the field differs. Therefore, the logical relation would be created using the field `Country_ID` from the `Countries` table and `Country` from the `Customers` table.

Some Terminology

Here are some terms you should know when it comes to relational databases:

Table
: The structure that stores the data inside a relational database.

www.it-ebooks.info

Fields (Columns)

The individual unit (or category) of information inside a table. Each column or field will have one data type; for example, integer, string or text, date, and so on.

Domain

The data type of a field or column. It may also include the list of possible values accepted. As an example, a field like `gender` could be a text field, but it might also be limited to two possible values: "male" and "female".

Record (row)

A set of columns (fields) that hold related information of different data types.

Primary Key (PK)

One or more fields that can be used to identify a record. Primary Keys need to satisfy three conditions:

- Uniqueness, that is, there can't be two records that have the same value on that particular field,
- The field should always have a value for each record, and can't be left empty.
- It must be an index field (see below).

Composite Key

A Primary Key that is formed by more than one field.

Subrogate Key

A field that was created solely to identify a record and has no other purpose; that is, to be a Primary Key. It may be because there is no other field that can be used or, even if there is, it may not be practical.

For instance, let's return to our table in Figure 11.1, the customers table. I can easily use the field `Customer_ID` to identify each record and make it unique. So even if I had two people named "John Doe" living in New York, and they were both 20 years old, I could still access each record individually because at least one piece of data would be different: their customer number. Apart from this, the field has no particular meaning and, outside this scope, it makes little sense. This is a subrogate key.

www.it-ebooks.info

I could have chosen a different route, though. Instead of creating an artificial field to do the job, I could have added a field named `Passport_Nbr`. Passport numbers are unique, so this new field would make a great Primary Key while still having a specific meaning. Bear in mind, though, that there would be more characters involved.

Index

This is a technique used to speed up searches. When a field is indexed, it is added to a table made up only by indexed fields. When we do a search, those fields are looked for first. Searches can be done on any field—indexed or not—but are faster on indexed fields. That's the only difference.

One table can have one or several indexed fields. As a matter of fact, it's a good idea to index any field that will attract a lot of searches. However, indexing too many fields will have the reverse effect, slowing down the searches instead of speeding them up.

Foreign Key (FK)

This is one or more fields related to the Primary Key from another table. It can be used to match a record or file in a different table. Primary Keys can't have duplicate values, but Foreign Keys can, and most of the time they do.

Normalization

A technique used to design databases that avoids the duplication of data and inconsistencies.

Structured Query Language (SQL)

The language used to extract and manipulate the information in a database. What's good about SQL is that you learn it once and then use it on every single database, as the basics are common to all. Yet, each database adds its unique sets of commands or clauses to the standard.

Types of Relationships

Most of the time, we create a relationship between two tables using the Primary Key of one table and the Foreign Key from another. By definition, they're perfect for the job.

www.it-ebooks.info

There are three types of relationships that we need to be aware of: **one-to-one**, **one-to-many**, and **many-to-many**.

One-to-One Relationships

This type of relationship exists when one record in one table can only be related to only one record in another table.

For example, take the `Country` table from Figure 11.2 again. Let's say that there is some other table with additional information about those countries: a `Country_Detail` table, shown in Figure 11.3.

TABLE 3: COUNTRY_DETAIL

Country_Data_ID	Country_ID	Capital_City	Country_Tel_Code	Continent
1	3	Washington D.C.	1	North America
2	2	Madrid	34	Europe
3	4	Berlin	49	Europe
4	1	London	44	Europe

Figure 11.3. Adding a `Country_Detail` table to the mix

These tables can be related to each other in a one-to-one relationship using the fields `Country_Code` from both tables (in this case, the name of the related fields is the same, but it doesn't have to be). As we can see, each record from the Country table can only be matched to one record from the `Country_Detail` table. It's impossible for Germany or any other country to have two capital cities, or belong to two different continents.

You could say that both tables work as just one table. Often, when a table is too large with multiple columns and becomes hard to maintain, the administrator may split it into two or more different tables, and creating a one-to-one relationship.

One-to-Many Relationships

This type of relationship happens when one record from one table can be related to more than one field from the second table. It's the most common type of table relationship.

Let's take our `Customers` table again, and now create a second table, because we've invoiced these people.

www.it-ebooks.info

Customer_ID	Name	LastName	City	Country	Age
1	Jamie	Smith	Houston	USA	24
2	Susan	Taylor	London	UK	35
3	Marta	Lopez	Madrid	Spain	41
4	Freddie	Schmidt	Berlin	Germany	28

Invoice_ID	Invoice_Date	Invoice_Concept	Invoice_Amount	Customer_ID
1	10-01-2011	Hosting	$120	3
2	10-22-2010	Web design	$550	2
3	10-23-2010	Hosting	$300	1
4	10-30-2010	WordPress Template	$1000	1
5	11-02-2010	Web Development	$2700	2
6	11-16-2010	Drupal Template	$5200	3

Figure 11.4. A one-to-many relationship between customers and invoices

You can see that some of these customers have been invoiced more than once, although Freddie Schmidt (customer number 4) has made no purchases yet.

A relationship between these two tables would be one-to-many. Each value in the `Customer_ID` field in the `Customers` table only appears once in that table, but it can appear as many times as needed in the related field from the `Invoices` table.

Many-to-Many Relationships

A record in one table can be related to many records from the second table, and vice-versa; a record from the second table can be related to many records from the first.

In our example, let's say that we have one table recording each sale and who handled it. If we have a table with the names of the salespeople in our company, it may be that each customer might speak to several salespeople when enquiring about projects and invoices. Similarly, a salesperson would provide service to many customers.

Many-to-many relationships are unable to be created directly, so we need to create an intermediary table of two one-to-many relationships. This intermediary table will have a compound Primary Key, formed by two Foreign Keys; these would be the Primary Keys from the two tables we want to relate in a many-to-many relationship.

www.it-ebooks.info

Customer_ID	Name	LastName	City	Country	Age
1	Jamie	Smith	Houston	USA	24
2	Susan	Taylor	London	UK	35
3	Marta	Lopez	Madrid	Spain	41
4	Freddie	Schmidt	Berlin	Germany	28

Customer_ID	Salesman_ID
1	3
1	2
2	2
2	1
3	1
4	3
1	3

Salesman_ID	Salesman_Name	Salesman_Last_Name
1	Jeremy	Gilford
2	John	Gartland
3	William	Turner

Figure 11.5. An example of a many-to-many relationship

Referential Integrity

Referential integrity is a database constraint that ensures that the references between data are valid. It simply means that the values in a Foreign Key will always match one value from the Primary Key in the parent table (the table they are related to). It also means that a Foreign Key field needs a value for each record; it should never be empty. This is logical, because most of the relationships between tables are created by matching the values from the Primary Key in the first table with the values from the Foreign Key in the second table.

More on Normalization

We touched on normalization earlier—the technique or process that avoids duplicates and keeps the consistency of data. In practice, it is a way of organizing information to avoid redundancies (that is, unnecessary repetition of information) anomalies,

www.it-ebooks.info

and errors. I should add that normalization tries to simplify the design of a database and intends to optimize it. The result is more robust and faster databases.

The goal is that any data addition or deletion will be made in just one table and then propagated through the database via relationships.

There are various rules or concepts known as **Normal Forms (NF)**. The first three norms are the most important, although there are six in total. As you can see, the world "normal" is common to databases!

The higher the Normal Form applicable to a table, the less vulnerable it is to inconsistencies and anomalies. If a table complies with the 4th Normal Form (4NF), it will comply with the three previous rules.

It's very rare to see a database that needs to be normalized up to the 5th Normal Form (5NF); in most cases, it's only necessary to meet up to 3NF. Yet, if the design of the database complies to 3NF, it's very possible that it complies with 4NF and 5NF.

First Normal Form (1NF)

- All the values in a column (field) of data should be atomic; that is, they can't be divided or split, and should only have one piece of information.

As an example, the field `Name` may be "Jason Alpert"; as this value can be divided in two (first name and last name), it is not atomic.

- No column (field) will be duplicated in a table.
- Separate groups of related information should go in one table each, and identify each row with a unique value (create a Primary Key).

The first part, separating groups of information, means that each table should only have fields that are directly related to the topic of the table.

As an example, in our database we now have four tables. The first is related by customers, and includes all the information about them (customer id, name, where they live, and it could also have email addresses, telephone numbers, and so on).

www.it-ebooks.info

The second table is related to countries and should have information directly involving them. That is, if I needed more information about the UK, I could have added fields like population, time zone, or flag.

The third is related to invoices, and holds all the information directly related to it such as invoice date, amount, and so on. It could even have information about payment details, payment date, and so on.

Second Normal Form (2NF)

In addition to meeting the requirements of the 1NF, the 2NF adds new constraints.

- Rows should be related to *all* of the Primary Key, not just part of it. This means that if I have a composite key, a record needs to be related to all those fields that form the composite key, not just one of them.
- Subsets of data that apply to multiple rows of a table should be placed in a separate table.
- You can create a relationship between these new tables using Foreign Keys.

In our database, I removed the countries from the Customers table and placed them into their own table. Then I created a relationship between the `Customers` table and the `Country` table.

Third Normal Form

As well as meeting the previous two forms, the third Normal Form goes a bit further. All nonkey columns need to be mutually independent.

Let's review the `Invoice` table once more and add some more fields (in violet).

Invoice_ID	Invoice_Date	Invoice_Concept	Invoice_Amount	Tax	Total	Customer_ID
1	10-01-2011	Hosting	$120	18%	141,6	3
2	10-22-2010	Web design	$550	18%	649	2
3	10-23-2010	Hosting	$300	18%	354	1
4	10-30-2010	WordPress Theme	$1000	18%	1180	1
5	11-02-2010	Web Development	$2700	18%	3186	2
6	11-16-2010	Drupal Template	$5200	18%	6136	3

Figure 11.6. Adding a few new fields to the `Invoice` table

www.it-ebooks.info

The `Total` field depends on two other fields: `Invoice_Amount` and `Tax`. This means that if I modify any of those fields for any record, I'm forced to change the `Total` field too. The field is related to the Primary Key indirectly, since its direct dependence relies on other fields. This is called **transitive dependence**. The `Total` field should have been left out and the amount calculated when querying the table.

Note that calculated fields is just one example; other situations exist, too.

Database Design

A good design starts with pen, paper, and some thinking. Your very first task should be to question why you are building a database and what is its goal. Which queries should it answer, and what data will be extracted from it, and how?

Further questions to ask include: Who is going to be using the database? How many people will require access, and what sort of roles will they occupy (for example, administrators, users)? What is their level of experience and what kind of work do they do? What kinds of questions will they seek answers for? Do they produce daily, monthly, and/or yearly reports?

Will there be any printed forms? If so, is there an example of how those printed forms should look? Who will be introducing data to the database, and how often? How much information will there be, and will the data be modified often?

All these questions (and any others you can think of) will help you figure out which tables you'll need and the fields that should go in each of them, as well as the relationships between tables.

The answers will also force you to make a decision about which brand of database to use. Not all databases have the same characteristics or capacity. Access is good for small offices with small-to-medium needs, while Oracle is expensive but is better at handling huge amounts of data. We'll look at database options in the next section. Other considerations include the user's operating system, as well as the server used.

Once we have our questions answered and our database drafted on paper with all its tables, fields, and relationships, we now go to our computer and create it.

www.it-ebooks.info

Which database to choose?

There are hundreds of databases out there, but a few are extremely popular:

- Oracle
- MySQL
- PostgreSQL
- SQL Server
- SQLite
- Access

This list could grow and grow. Some databases are operating system-dependant like Access or SQL Server, which only work with Windows. Others will perform better in one OS than in others. MySQL was first designed for Linux servers, although it does perform well under Windows. Some can hold and quickly retrieve large amounts of data, while others are quite limited in that aspect. In the above list, Access is the database with the smallest capacity.

Know that good database design has a high impact on capacity.

Access is also the only one listed above with a graphical interface. Although it's possible to install additional software on some systems to enjoy the advantages of a GUI, it won't be as sophisticated as Access's. Be warned that while a graphical interface is easier to use, changing to another database becomes harder down the track.

Some databases are free; others are expensive. All of them have their advantages and disadvantages. My suggestion is to opt for MySQL from the very beginning, because it:

- performs well under Linux and Windows
- has a free and paid versions, so you're not forced to pay a lot of money for the license
- is included with no extra cost in many, if not all, hosting plans
- can handle large amounts of data
- is a natural "partner" if you develop with PHP

Furthermore, the list of CMSs (Content Management Systems) and applications based on MySQL is huge; it includes, but is not limited to, these very popular soft-

www.it-ebooks.info

wares: WordPress, Joomla, Drupal, Magento, b2evolution, osCommerce, PHP-Nuke, and vBulletin.

If you develop with ASP (Classic or .NET), you still can use MySQL. In this case, you may want to go all the way with Microsoft, so your database is more likely to be SQL Server—Microsoft's own database, which happens to not be free.

MySQL and SQL Instructions

SQL or Structured Query Language is common to all databases; although every database extends or modifies it somehow, the basics are the same for every database. They all follow the ANSI-92 standard. Although a more recent standard now exists, ANSI-92 is still the most popular.

What I'll list here are the most commonly used SQL instructions and their basic implementation. These should work well in all databases, but I will base their function on MySQL. Why? Because while all database share a common ground (such as this standard SQL), there are variations. As an example, all databases use different data types: numeric, dates, text, Boolean … but the name they give to each data type may differ. In Access, a text field is defined as `text` (up to 255 characters) or `memo` (for larger texts), but in MySQL these data types are `char`, `varchar`, or `text`; even with text we can enjoy the benefits of choosing its capacity, since we have `tinytext`, `text`, `mediumtext`, and `longtext` (from smaller to larger).

It's wise to remember that MySQL has no graphical interface on its own, so you'll have to write everything using a text interface.

Some Good Practices

Although some databases allow you to define names for fields or tables using spaces, it is much better to avoid using them. For example, instead of naming a field `Customer Name`, change it to `CustomerName`.

Try to also be consistent with your naming convention. All names should be descriptive and clear but avoid making them unnecessarily long, and don't start them with a number or symbol—the exception being the underscore. If you always write in lowercase, don't start using capital letters. In my case, I always write in lowercase except when I choose a name that needs two words; then I capitalize the first letter

www.it-ebooks.info

of the second word. Underscores are harder to write when you use non-English keyboards!

Note that it's better to name each field uniquely. This means that if you have two tables—one for customers and one for suppliers—it's likely that they'll share common information such as name, last name, address, and telephone, for starters. Instead of calling the field `name` in both tables, consider naming them `customerName` (or `Customer_Name` or `cust_name`) and `supplierName` instead. This way, you'll know exactly in which table it belongs and there will be no ambiguity.

If you're adding a Foreign Key in a table, it's common practice to name the field after the Primary Key in the parent table. It's then easier to remember the names of the fields that will be used to create a relationship between those tables. I normally add FK on the second field to indicate that it's a Foreign Key.

For instance, the first two tables at the beginning of the article—`Countries` and `Customers`—have a field named `country_ID`. In actuality, I'd have named them `countryId` as Primary Key of the `Country` table, and `countryIdFK` in the `Customers` table, to remember that it's a Foreign Key, as well as make it a different name.

Most databases have no qualms whether you write all-caps or all-lowercase. They don't really bother about those things for SQL instructions. However, they do mind very much when it relates to names of databases, tables, or fields: `customerName` is not the same as `CustomerName`. Mind, this does depend on the database you use as they all behave differently.

Note: All SQL instructions end with a semi-colon so the computer knows that the instruction is complete

SQL Basic Syntax

Be aware that this list is extremely limited, and you should always look at the manual of the database you're using.

Creating a Database

Nothing could be easier. After receiving the welcome message upon logging into MySQL, create a new database by simply writing:

www.it-ebooks.info

```
CREATE DATABASE myNewDB;
```

In MySQL, if you want to work with this database, you need to write **`USE nameOfDatabase;`** so the system knows it.

Creating a Table

This takes a bit longer as you need to list the name of the fields and data types. As an option, you may indicate which field will be the Primary Key or Foreign Key.

```
CREATE TABLE nameOfTable (fieldName1 data_type [attributes], fieldName2
data_type [attributes], … , [CONSTRAINTS])
```

As an example, let's create the `customers` table with three constraints: a Primary Key, a Foreign Key, and let's index the last name as it's likely that customers will be searched by this field. Constraints are optional:

```
CREATE TABLE customers (
  customerID INTEGER NOT NULL UNIQUE,
  customerName VARCHAR(25),
  customerLastName VARCHAR(25),
  customerCity VARCHAR(50),
  countryIDFK INTEGER, customerAge INTEGER,
  PRIMARY KEY(customerID),
  FOREIGN KEY(countryIDFK) REFERENCES country(countryID),
  INDEX(customerLastName)
);
```

This syntax varies a little from database to database, and it's worth noting that some databases, like MySQL, can use different storage systems that may affect the syntax.

Observe that when I create a Foreign Key, I need to specify the name of the parent table and the associated field. Of course, the table `country` needs to exist before I create the table `customers`; otherwise, I'd be unable to reference any of its fields.

Changing the Structure of a Table

Altering the structure of a table to add or delete fields, or change data types, is relatively simple in its basic form. Be aware that there may be unwanted or nasty consequences, especially if the database is already in use.

www.it-ebooks.info

To change the data type of a field, we could use the following syntax. We could take advantage of having to do this and change its name too, but this is merely optional.

ALTER TABLE *nameOfTable* ADD/MODIFY/CHANGE/DROP *nameOfField* [newNameOfFieldIfIAmChangingIt] INTEGER;

The instruction ALTER TABLE can be very complex, so check the manual accordingly:

```
ALTER TABLE customers MODIFY customerLastName VARCHAR(40);
```

In this case, I modified the capacity of the `customerLastName` field, which could only store up to 25 characters before this change.

Inserting Data into a Table

To add new data into a table, use INSERT INTO, which can have two different syntaxes:

INSERT INTO *nameOfTable* (*field1*, *field2*, *field3*, …) VALUES (*value1*, *value2*, *value3*, …)

INSER INTO *nameOfTable* SET (*field1=value1*, *field2=value2*, *field3=value3*, …)

For example:

```
INSERT INTO customers(
  customerID,
  customerName,
  customerLastName,
  customerCity,
  countryIDFK,
  customerAge)
VALUES (1,'Jamie','Smith','Houston',3,24);
```

The important point here is that you write the values in the same order that you listed the fields. Notice also that while text values need quotes, numbers do not. I tend to use single quotes but double quotes will work too.

www.it-ebooks.info

Modifying a Record

If, instead of changing new data, we need to modify existing records, there is another instruction.

```
UPDATE nameOfTable SET field1=value1, field2=value2 WHERE condition;
```

Of course, if you wish to change every single record, there's no need for conditions; however, even in that case, some databases do need a `WHERE` clause. In those cases, your condition needs to always evaluate to true; most of us tend to use 1=1.

Here's an example of modifying a record:

```
UPDATE customers SET customerName='James' WHERE customerID=1;
```

Reading Data

Of course, none of this really matters if we're unable to query the database and see the information that's inside. By far, the most important and used instruction in SQL is `SELECT`. This instruction allows you to read what is inside one table, or two, or three. Relationships among tables are important if you want to extract information from more than one table in just one query.

You don't need to extract the information from all fields, just the ones you need, and of course you can filter the result by specifying conditions. There are some additional clauses to order the results and more:

```
SELECT field1, field2, field3 FROM nameOfTable [WHERE condition];
```

```
SELECT customerName, customerLastName, customerCity FROM customers;
```

In the example above, I chose to view all the records from the table customers, but I wanted the name of the customers and the city where they lived.

If you want to obtain results from more than one table, it changes a bit.

```
SELECT t1.nameOfField, t2.nameOfField FROM table1 AS t1 INNER JOIN
table2 AS t2 ON t1.nameOfRelatedField = t2.nameOfRelatedField;
```

www.it-ebooks.info

This particular syntax uses an alias to name the tables. The reason for this is that if you query more than one table, you need to not only list the fields, but indicate which table they belong to. In order to do so, you'll write **`nameOftable.nameOfField`** when listing fields. This can take a long time, so shortening the name of the table may be a good idea. To use an alias for a table, we use the clause `AS aliasName`:

```
SELECT t1.customerName, t1.customerLastName, t2.countryName
FROM customers AS t1
INNER JOIN country AS t2
  ON t1.countryIDFK=t2.countryID
WHERE t1.customerID<=3;
```

There are different types of joins or relationships, but `INNER JOIN` is the most common. It simply means that it extracts data from the second table when the values in the related fields match.

www.it-ebooks.info

Chapter 12

The Iceberg of TCP/IP

by Robert Wellock

Hmm, I'm sure most of you who have decided to read this chapter will have heard strange rumors about the Internet being controlled by a mysterious dark force know as TCP/IP …

That's only part of the story. TCP/IP is an immense topic, which is why I refer to it as the Iceberg. If you have a "brain the size of a planet" like Marvin the Paranoid Android,[1] you might just be able to memorize the whole of the TCP/IP protocol stack. Rather than attempt that amazing feat, I'm going to write about what is just the tip of the topic: a selection of protocols focusing on what is most relevant to web design or general home computing. So let's dive in, shall we?

What is TCP/IP?

Transmission Control Protocol/Internet Protocol (TCP/IP) is the de facto flexible network communication protocol (or a set of rules) that helps form the basis of the Internet. TCP/IP determines how network-capable devices should connect to the

[1] From Douglas Adams' *The Hitchhiker's Guide to the Galaxy*: http://www.hhgproject.org/entries/marvin.html

www.it-ebooks.info

Internet, whether they're a computer, games console, security camera, or web-enabled electronic fridge. Your web browser, internet server, and email client all use TCP/IP, as does every public web address—domain name—on the World Wide Web. The Internet is based on the client/server model: when a server is running it waits for a client request; for example, a web browser asking for your favorite website.

Brief History of TCP/IP

Computer systems were developed in Great Britain during World War II. Later military research by the US agency DARPA (Defense Advanced Research Projects Agency) produced basic networks that eventually evolved into the Internet. The Transmission Control Protocol/Internet Protocol network model was originally based upon the OSI (Open Systems Interconnection) model, but eventually became the most prominent networking suite in use today.

These networks needed a reliable and robust method for data to be transmitted in the event of major communications damage or nuclear holocaust. If you've seen the sci-fi film *Terminator*, Skynet was the fictitious defence network that was connected via TCP/IP which, as a result, couldn't be effectively shut down.

In the 1980s, only a few hundred computers were connected by the Internet. By 1997, there were around 16 million computers, and growth has been exponential ever since. However, it was Sir Tim Berners-Lee who was primarily responsible for its explosive growth when he conceived of HTML (1989), with the breakthrough World Wide Web following in the 1990s.

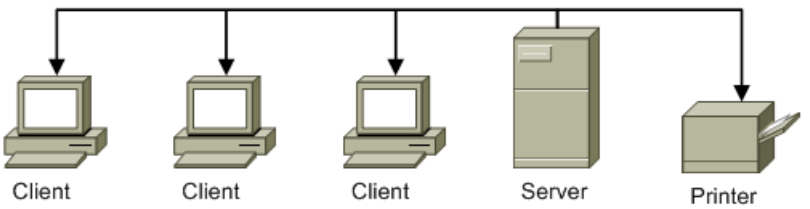

Figure 12.1. Client/server network model

As you can be see, the network server is at the heart of this model, distributing services to clients (computer workstations) and network devices through one central point.

www.it-ebooks.info

The OSI Model Compared to the Internet Model

There are seven layers in the Open Systems Interconnection (OSI) model, which was developed in 1984 by the International Organization for Standardization (ISO). Note that we have an OSI created by the ISO. The OSI model handles various tasks via an international protocol. The OSI is built upon public **open standards**, meaning it is not proprietary and free to use, similar to open source software like the Linux operating system.

The OSI model divides networking functions into seven layers; confusingly, it numbers from bottom to top in descending order (instead of top to bottom in ascending order). The OSI model begins with the Physical layer at the base of the stack.

Each layer has specified roles and functions that depend on the layer above or below. Data is passed down the OSI reference model layers, and each layer will prepare and package the data for network transmission. You can think of it like an assembly line.

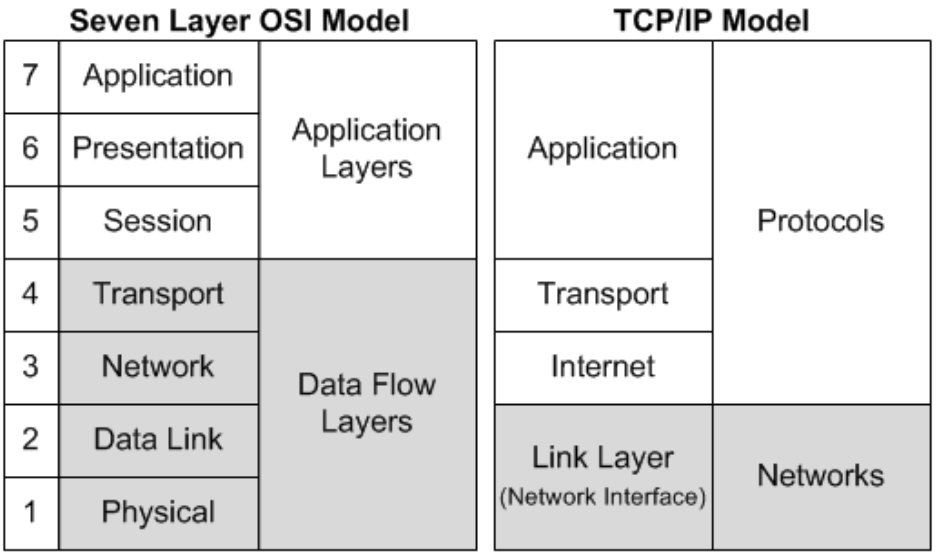

Figure 12.2. Comparisons of the OSI and TCP/IP model

www.it-ebooks.info

Remembering the OSI Layer using Mnemonics

Networking technicians use mnemonic phrases to help them remember the layer order of the OSI. Two popular ones are All People Standing Totally Naked Don't Perspire (going from Application Layer 7 to Physical Layer 1), or the more common Please Do Not Throw Sausage Pizza Away (Physical Layer 1 to the Application Layer 7).

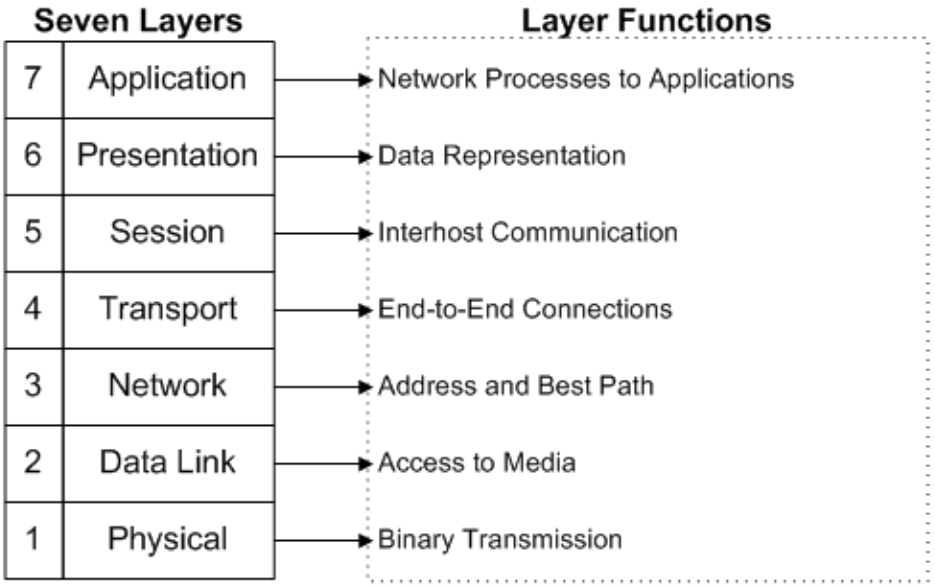

Figure 12.3. Seven-layer OSI reference model and functions

Layer 7: the Application Layer

This layer describes how the applications talk to each other. Network processes to applications include file transfer, access and document management, and message interchange. Application protocols that sit at this layer include email, FTP, and Telnet.

Layer 6: the Presentation Layer

This layer can translate one format into another, so that the application layer can communicate seamlessly with other operating systems, for example. Data representation: translation, transformation, and security (encryption of data).

www.it-ebooks.info

Layer 5: the Session Layer

This layer determines how the rules are applied when two applications want to speak, such as whether they take turns or send and receive data at the same time. It includes Interhost communication, and dialogue and synchronization control of network cards.

Layer 4: the Transport Layer

As its name suggests, the Transport Layer provides a delivery service for data; it can be either guaranteed delivery, or an unreliable best-effort delivery. It also segments data into more manageable packets. End-to-end connections and transfer management (connections, error-flow control, and segmentation).

Layer 3: the Network Layer

The main function is to determine how packets and data are logically directed from one network address to another. Address and best path involves network and routing addressing to correct stations. Routers operate at this layer.

Layer 2: the Data Link Layer

This layer is focused more on hardware. It specifies how packets of data are organized into frames for placing onto a network medium, but doesn't make decisions like Network Layer 3. A switch would be an example where it sends network data based on a Media Access Control (MAC) address,[2] but cannot forward it to another network. Includes access to media, deals with framing, data transparency, and error control during transmission.

Layer 1: the Physical Layer

This layer determines how binary data is converted into electrical pulses, fibre-optical signals, or radio waves (such as Wi-Fi) and transmitted between systems. Binary transmission, mechanical and electrical network interface definitions. This is your hardware cabling and media; for example, a modem cable.

The TCP/IP Network Model

The TCP/IP model is similar to the OSI model in design, though it only contains four layers. Some people also refer to it as the "Internet Model." The four layers of

[2] A MAC address (or MAC-Layer address) is a 6-byte (48-bit) long, globally unique identifier controlled by the IEEE that's physically burnt into networking device firmware chips; it's also known as a "hardware address" and is requirement for any networking device.

www.it-ebooks.info

the TCP/IP Network Model are the Application Layer, the Transport Layer, the Internet Layer, and the Link Layer (sometimes called the "Network Interface Layer" in networking books). Usually when people mention TCP/IP, they're actually referring to the TCP/IP model, since it's the most prominent protocol stack and less compartmentalized than the whole OSI model.

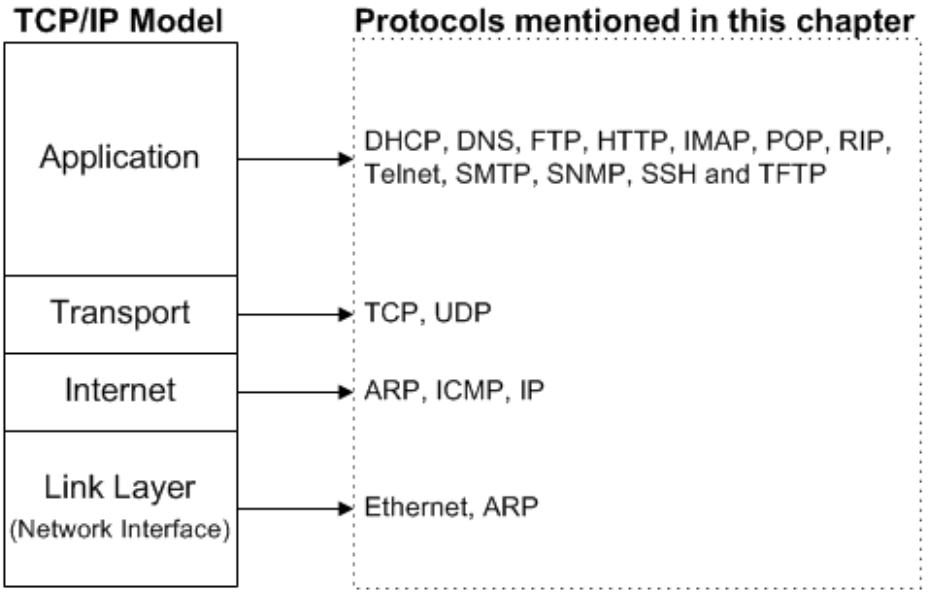

Figure 12.4. TCP/IP protocols—the "Internet Model"

If you cast your mind back (or look at Figure 12.2), you'll notice that the TCP/IP Model is split at layer 3 and 4 of the OSI Model. OSI uses layers called "Transport" and "Network," whereas the Internet Model uses "Transport" and "Internet." Despite having similar names and functions, they do differ; the reason they're split at that level is to do with TCP and IP residing at those layers.

TCP: A Connection-orientated Protocol

Transmission Control Protocol (TCP) is a connection-orientated protocol (Transport Layer 4)[3] that involves **handshaking**: a computer requests a connection with another foreign device, like a modem or printer, and requires it to respond. The two will

[3] This rest of this chapter will refer to the OSI Model Layer numbers and names, rather than named TCP/IP layers, to prevent ambiguity.

www.it-ebooks.info

agree upon a set of rules for communication until it is closed or terminated by one of them. TCP identifies applications using it (such as FTP or email) via assigned port numbers.

The TCP protocol is full-duplex (that is data is transmitted simultaneously in two directions), and when it receives data from the upper layers of the OSI, it will guarantee reliable delivery to other networks. Thus, a TCP acknowledgment overhead will consume some bandwidth, making it useful for transmitting large amounts of data in a reliable fashion. If TCP were a car, it would be a Rolls-Royce, guaranteeing plenty of features and a high-quality, reliable service.

This protocol works in conjunction with the IP (Network Layer 3), and has the role of *guaranteeing* that messages arrive at their destination, or notifying the application programs that there's been a failure if they can't be delivered.

IP: A Connectionless Protocol

Internet Protocol (IP) is considered a connectionless (Network Layer 3) routable protocol, and uses a "best-effort" delivery mechanism that's can't guarantee message delivery, unlike TCP. Within the IP protocol, the data is encapsulated into packets and sent via the Internet. It can take various paths via routing protocols to reach its final destination.

Because IP is a connectionless service, it's unreliable,[4] hence why it's unable to guarantee the delivery of data (so it can be lost), or the order it was sent. This is why it usually works alongside the connection-orientated protocol TCP, which enables the dependable and orderly deliver of those data packets.

Routers

Routers, which most of you will be familiar with, are forms of hardware that send data from one network segment to another, and operate at IP level (Network Layer 3). They are an essential aspect of the Internet as they direct traffic and data across the globe via various networks.

[4] Do not confuse the term "unreliable delivery" with poor service. On a correctly working network it should not be an issue.

www.it-ebooks.info

Figure 12.5. Router connections to the Internet cloud

Routers discover paths to different networks and learn how to forward packets to their destination via other routers. Two main protocols they use are Routing Information Protocol (RIP) and Address Resolution Protocol (ARP). ARP is used to find a MAC (hardware) address on a local machine when its IP is known. RIP works using a metric that's distance-based and hop-count-based to choose the best path to a destination.

IP Addressing and Subnets (Subnetting)

Within a TCP/IP environment, stations communicate with servers, hosts (network hosts), and other end stations. This is achieved because each network node has a unique 32-bit logical address, known as an IP address (consisting of IP numbers). This 32-bit binary address (0s and 1s) is divided into four groups of 8 bits called octets.[5]

One octet can contain 256 (or 2^8) different values represented in binary. For example: 00000000, 00000001, and 00000011, going up to 11111111. A single computer byte usually contains 8 bits and TCP/IP uses four octets (4 bytes).

Binary Numbers and the Base-2 System

Since binary is a base-2 system, each digit represents an increasing power of 2, with the rightmost digit representing 2^0, the next representing 2^1, then 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, and so on.

Thus, the decimal 42 would be 00101010 in binary notation:

$[(0) \times 2^7] + [(0) \times 2^6] + [(1) \times 2^5] + [(0) \times 2^4] + [(1) \times 2^3] + [(0) \times 2^2] + [(1) \times 2^1] + [(0) \times 2^0] = 42$

[5] A bit equals 1 or 0 in binary, and the 32-bit address allows up to 4,294,967,296 (2^{32}) possible unique addresses.

www.it-ebooks.info

Table 12.1. Binary Numbers and the Base-2 number system

Position power	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	128	64	32	16	8	4	2	1
Example: 42	0	0	1	0	1	0	1	0

The bit farthest right is known as the **least significant bit (lsb)**, while the bit on the far left is known as the **most significant bit (msb)**.[6]

A 32-bit IP address can be expressed as four groups of decimal numbers using the range 0 through 255 separated by dots: 172.16.0.1, for example. Converting this address into 32-bit binary notation would be: 10101100.00010000.00000000.00000001. As you can see, it's far easier for a human to read, remember, or type the decimal version. However, computers only use binary, so they'll use the 32-bit binary notation, or convert the decimals into binary.

For network communications to be successful, each network and machine connected must be able to identify one another on the network segment, or on a wider scale of other networks. Think of how a telephone system works by assigning individual telephone numbers to a premises, allowing people within to contact each other without complications.

As with international telephone numbers, every IP address hosted upon the public internet must be completely unique—like a fingerprint. No duplicate IP numbers are allowed or the entire internet would cease to work; it would be unable to route traffic; that is, data packets from sender to receiver.

IP addresses for the globalized Internet are usually assigned by Internet Service Providers (ISPs) and large telecommunications companies that lease ranges of IP addresses to businesses or individuals.[7]

[6] MSB (in all capitals) stands for Most Significant Byte. Sometimes "high-order bit" is used instead, but they are the bit(s) with the highest value.

[7] There are also private networks (private IP addresses not directly connected to the Internet) that can have their numbers assigned by network administrators (network working group RFC 1918).

www.it-ebooks.info

Addressing with IP Numbers

The IP addressing system is built upon a hierarchy, with every IP address containing two parts: one that identifies the network portion and one that identifies the host portion.[8]

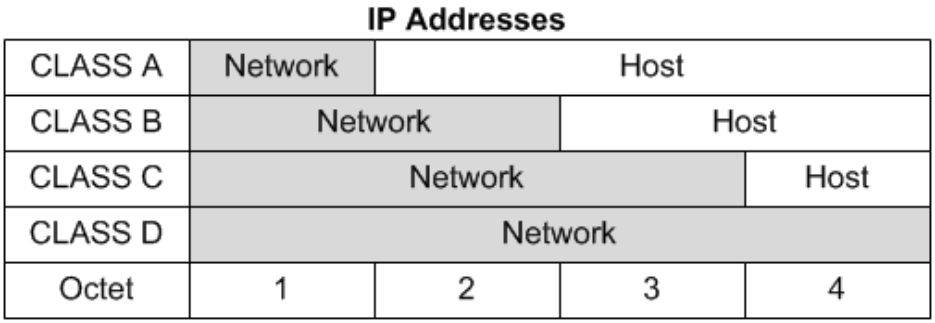

Figure 12.6. IP addresses showing network and host portion

All available IP addresses are divided into classes of various sizes, ranging from the largest (Class A) to medium (Class B) to small (Class C). There are two other special classes: Class D and E .

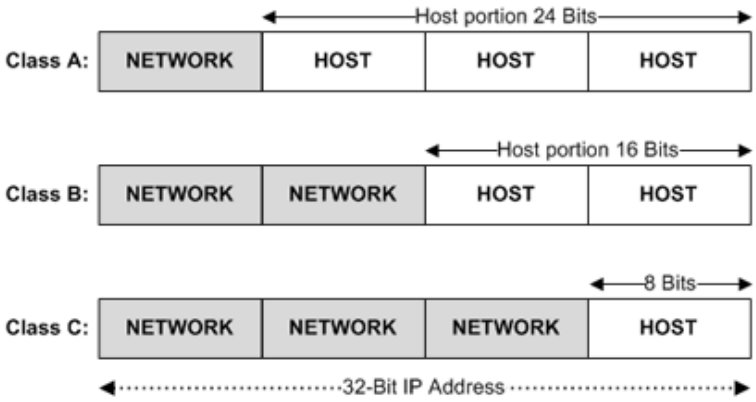

Figure 12.7. Class A, B, and C addresses

[8] Throughout this chapter "Hosts" generally refers to Network Hosts, not Web Hosting Providers. In networking, host usually refers to a range of PC(s) or Router(s) assigned an IP address; that is, computer(s)/workstation(s) connected to a network.

www.it-ebooks.info

Dividing these addresses and classes into usable networks, usually referred to as subnetting or subnetworking, is a complex, in-depth topic—but impossible not to mention in a Chapter on TCP/IP!

The Complexity of Subnet Calculations

Subnet calculations are intricate and really beyond the scope of this chapter. Any descriptions here will be kept brief, so don't worry if you struggle to fully understand this section—I've just added it for completeness. I'd suggest searching on the Internet if you want to fully understand subnet calculations.

Table 12.2 shows the classification of IP ranges, and how many networks and hosts can be achieved through subnetting.

Table 12.2. IP Address Classes

IP Address Class	# of Networks	# Hosts per Network
A	126[a]	16,777,216
B	16,384	65,535
C	2,097,152	254
D (multicast)[b]	–	–

[a] The 127.x.x.x range of addresses is reserved for loopback addresses, used for the purposes of testing and diagnostics only.

[b] Multicast is a unique network address used for directing packets to specific groups of IP addresses.

Table 12.3 uses both binary and decimal notation to show how the address is identified.

Table 12.3. Identifying Address Classes

IP Address Class	High Order Bits	First Octet Address Range	# of Bits in the Network Address
A	0	0–127[a]	8
B	10	128–191	16
C	110	192–223	24
D	1110	224–239	28

[a] The value 127 (01111111), a Class A address, is reserved for a loopback address used for testing and diagnostics, and can't be assigned to a network.

www.it-ebooks.info

So 212.183.140.37 is a Class C address, as it starts with 212 (falling between 192–223). In binary, a Class C address always begins 110[…] (using high order bits). The decimal 212 would be written: 11010100 in binary.

Class A, B, and C Addresses

Within Class A addresses, the first number (octet) value is the Network portion; the last three, the Host portion (as seen earlier in Figure 12.6). Networking professionals sometimes use the format N.H.H.H, where N is the Network and H is the Host (local network devices, computers, and so on). You'll notice that only 1 to 126 (Table 12.3) is a valid Class A network because 127.0.0.1 is a reserved local loopback[9] when testing a local NIC (Network Interface Card).

The American Registry for Internet Numbers (ARIN)[10] is one of the regional internet registries that assign global network numbers, the N bits in Table 12.4. The host portion (H) is defined by a network administrator.

Table 12.4. IP Address Classes

	8 bits	8 bits	8 bits	8 bits
Class A	N	H	H	H
Class B	N	N	H	H
Class C	N	N	N	H

Subnetting

The process of subnetting helps to address the limitation of global IP addresses and the prospect of them running out. It splits a "network portion" of an IP address into smaller subnetworks on the same network.

[9] The loopback device (also known as "localhost") is a virtual network interface implemented in software only, and not connected to any hardware. If you're familiar with Apache HTTP Server, PHP, and MySQL, you may have used http://127.0.0.1 or http://localhost in the browser location bar as the local test server settings.

[10] http://www.arin.net/

www.it-ebooks.info

Subnetting designates bits from the host portion and groups them with the network portion, and uses bitwise AND[11] calculations.

As well as dividing an address into host and network portions, it can be further divided into a subnet number. A subnet mask uses a 32-bit number and dotted decimal like an IP address.

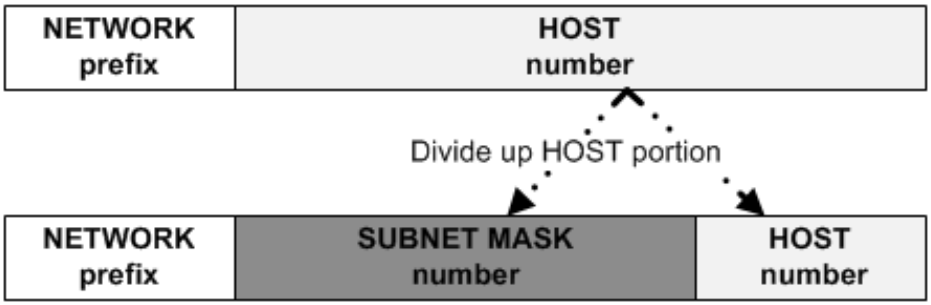

Figure 12.8. Creating a subnet by dividing the host identifier

You write a subnetwork mask in binary as simple 0s and 1s, or convert a decimal into binary. Next, identify the address's Class. Eliminate the 1 bits from the corresponding network part of the address. Any remaining 1 bits in the mask indicate that bits are the subnet portion of the address.

Table 12.5. Default subnetwork masks

Class	Binary Network Mask	Decimal Mask
A	11111111.00000000.00000000.00000000	255.0.0.0
B	11111111.11111111.00000000.00000000	255.255.0.0
C	11111111.11111111.11111111.00000000	255.255.255.0

Let's take address 212.183.140.37, a Class C address. The default subnet mask for a Class C address would be 255.255.255.0. These subnets help identify all computers on a 255.255.255.0 network. All hosts on this network have an IP address of

[11] The AND operation uses Boolean logic and takes two input values: 0 or 1. If both input values are 1, the logic gate will output 1; otherwise it outputs 0. Since network professionals think in binary numbers, they will AND IP addresses and subnet masks rather than use decimal.

www.it-ebooks.info

212.183.140.x, so they share the first three octets, but the "x" is the host portion. The IP address 212.183.140.37 belongs on network 212.183.140.0.

The maximum amount of usable hosts on a Class C address is 254 (as evident in Table 12.2). You're allowed 254 hosts on the network (212.183.140.x); that is, from 212.183.140.1 to 212.183.140.255 (remember that "x" is the network).

To calculate the amount of hosts on a specific network, you use a simple formula: 2N – 2 (where N in this instance equals the amount of usable bits for the host addresses). The value 0 (.0000000)—all 0s—is a broadcast address, so it can't be used for a host machine, and neither can 255 (.1111111)—all 1s. A Class C network uses a default subnet mask (255.255.255.0), and only the last octet (8 bits) "N" are available for hosts. Therefore $2^8 - 2 = 254$ hosts on a Class C network.

Table 12.6. Subnetting Chart

First Octet Decimal Notation	Number of Available Subnets	Class A Hosts per Subnet	Class B Hosts per Subnet	Class C Hosts per Subnet
192	2	4,194,302	16,382	62
224	6	2,097,150	8,190	30
240	14	1,048,574	4,094	14
248	30	524,286	2,046	6
252	62	262,142	1,022	2
254	126	131,070	510	—
255	254	65,534	254	—

You can have more than the standard Class A, B, or C addresses but this should just give you a taste.

Port Numbers

Port numbers are essentially a method of keeping track of all the different types of messages taking place within a network. Some ports are assigned specific tasks, but only some ports are reserved for specific use; most are assigned and released dynamically as the need arises. This way, the multitude of different protocols can be separated and handled more efficiently.

www.it-ebooks.info

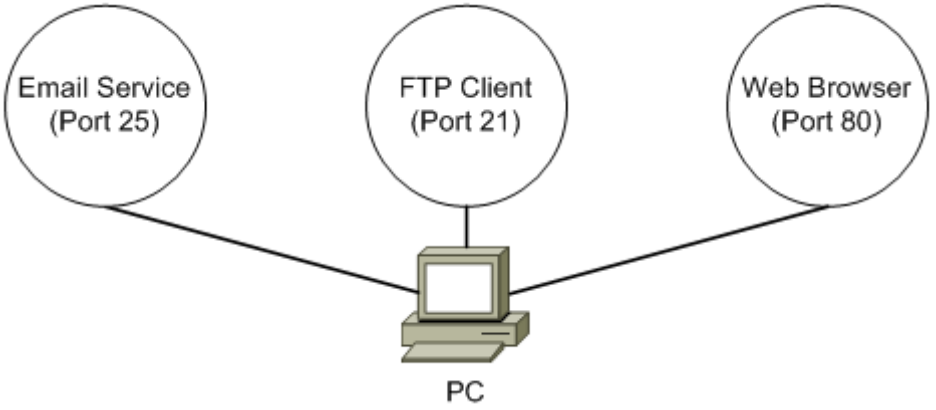

Figure 12.9. TCP port numbers in use by a PC in one instance

I'm sure many of you are familiar with the printer port; well, these are logical ports. Special well-known ports (RFC 1700) are reserved by the Internet Assigned Numbers Authority (IANA)[12], so that there's no ambiguity when connecting networks; for example, via the Internet. Some of the most well-known ports numbers include FTP (21), Telnet (23), SMTP (25), DNS (53) and SNMP (61), HTTP (80), and POP3 (110), hence why their associated protocols appear within this chapter.

Handshaking

Handshaking is an important part of a base operating system, because it determines how devices will exchange information between themselves. Two modems perform a handshake when one initiates communication with another (just as two people shake hands to greet each other) in order to exchange information, such as transfer protocol and speeds. The two devices then send messages back and forth.

The Different TCP/IP Protocols

TCP/IP communication involves a collection of different protocols.[13] I'm now going to discuss the most common ones in order to give you a brief insight into what happens behind the scenes when your computer connects to the Internet.

[12] http://www.iana.org/assignments/port-numbers

[13] Sometimes grouped communication protocols that operate together are referred to as protocol suites or protocol stacks. The protocols in this chapter belong to the TCP/IP protocol suite.

www.it-ebooks.info

User Datagram Protocol (UDP)

UDP[14] is a *connectionless* "unreliable delivery" protocol operating at Transport Layer 4. As a result, UDP has no requirement for receiving protocols like TCP (also Layer 4) to acknowledge the receipt of data packets. It doesn't concentrate on establishing connections like the TCP protocol, so it can transmit information faster than TCP; it is the upper application layers that are used to control its reliability. UDP is also useful for VoIP (Voice over Internet Protocol), streaming multimedia, and online multiplayer games moving small quantities of data—it's built for speed.

Domain Name Services (DNS)

Domain Name Services (DNS) handle the translation of an internet name (such as my own little website: www.xhtmlcoder.com) into an IP address.[15] Web browsers, electronic-mail applications, and FTP clients can use DNS. DNS protocols enable these clients to make requests to DNS servers that do the translations. Without DNS lookup servers, the Internet would be nearly impossible to use.

In other words, DNS is a name-resolution service for associating IP addresses with host names, making life much easier for humans.

Internet Control Message Protocol (ICMP)

Internet Control Message Protocol (RFC 792) handles error and control messages; for example, PING, unreachable, and timeout. Packet InterNet Groper (PING) sends ICMP messages to verify connections to a remote host. As covered, IP is an unreliable method for delivering data, and won't be delivered if a network communication fails because of faulty hardware. Instead, ICMP sends error messages to the sender informing of the delivery failure. Hence, it is an error reporting protocol for IP.

[14] The following protocols mentioned within this chapter: DHCP, SNMP and TFTP all use UDP (RFC 768); DNS can use both UDP and TCP. If UDP were a car it would be a lightweight Formula One racing car.

[15] Note that even though we have human-readable names in a web address, the computer uses an associated IP address to transfer traffic, not a name like www.xhtmlcoder.com. You could also use an IP value (if you knew it) to reach the same site (for example, 94.136.40.100), but bear in mind that IP numbers can alter .

www.it-ebooks.info

Routers use headers and a forwarding table to determine where packets go. They use ICMP to communicate with each other and configure the best route between any two hosts.

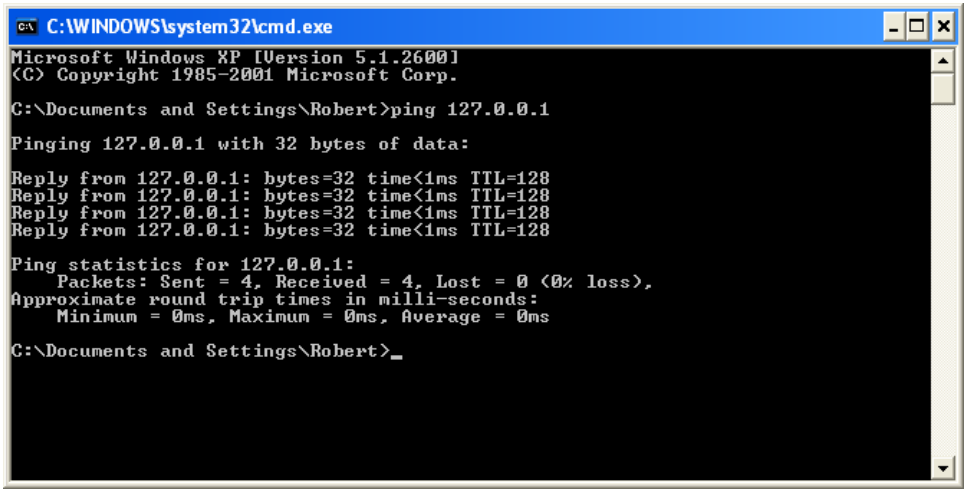
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Robert>ping 127.0.0.1
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Ping statistics for 127.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\Robert>

Figure 12.10. ICMP ping command testing TCP/IP

The PING program is probably the most well-known ICMP tool, and works at the TCP/IP Internet Layer.[16] Most of you will have used it to test your home network or internet connection (such as `ping 127.0.0.1` (loopback/localhost address) to see if TCP/IP is working. Try it if you want.

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is an error-free, session-orientated protocol that uses TCP ports 20 and 21.[17] It uses a client/server model, where clients open an (Application Layer 7) session with the server, authenticate, and then allow actions like uploading and downloading files; it will eventually time out if inactive (idle) for a long period of time.

Most of you will have used this file-sharing protocol, which usually requires the user to log in before transferring files. You'll probably have uploaded and downloaded multiple files to your website also, using FTP to connect to a remote server that may be using a different operating system.

[16] This is not an OSI Reference Model Layer.
[17] The standard port 20 is for data, while port 21 is for control.

www.it-ebooks.info

Other similar protocols include TFTP (Trivial File Transfer Protocol) and SFTP (SSH File Transfer Protocol).

Hypertext Transfer Protocol (HTTP)

If you've ever used the World Wide Web, you will have used both TCP/IP and HTTP protocol to surf via a browser. It uses a client/server model.

The familiar HTTP protocol uses TCP port 80, and allows clients to deliver documents authored in HTML and other markup languages to be rendered by a user agent such as a web browser. Typically, a web browser will send HTTP requests to a web server; the server will then use HTTP to send the data back to a browser in the form of a web page or type of file requested.

HTTPS (Hypertext Transfer Protocol Secure) is a secure version of the protocol. Typical uses include secure banking or online payment services where private or sensitive data needs protecting during transfer via a website.

Wireless Security and HTTP

There is still vulnerability of eavesdroppers if you use HTTPS via a Wi-Fi (802.11b wireless) connection without using encryption. An example of encryption would be Wi-Fi Protected Access (WPA) security protocol.

Dynamic Host Configuration Protocol (DHCP)

Dynamic Host Configuration Protocol (DHCP) uses a client/server based model, enabling network hosts (DHCP clients) within an IP network to (usually automatically) obtain their configurations from a (DHCP) server; for example, IP address (the most common for SoHo—small office/home office), subnet mask, and default gateway. Typically, the DHCP client is available in most mainstream operating systems, including Windows, Solaris, Linux, and Mac OS.

Simple Network Management Protocol (SNMP)

Simple Network Management Protocol (SNMP), TCP port 161, is a set of software protocols for managing complex networks based upon a database structure. The first standard of SNMP was developed in 1989. It's an Application Layer protocol

www.it-ebooks.info

that's enabled in most network devices such as routers, switches, hub printers, and servers.

SNMP works by sending messages, called Protocol Data Units (PDUs), to different parts of a network. SNMP-compliant devices, called agents, store data about themselves in a Management Information Base (MIB), and return this data to the SNMP requesters. It allows statistical checking of network activity; for example, benefiting a network by monitoring where bottlenecks are occurring, troubleshooting how applications are being used, and so on.

Telnet

Telnet is usually referred to as a terminal emulation program, which enables you to run interactive commands on a remote Telnet server. Please be aware that many people often confuse the Telnet protocol with the telnet client.

No data will transfer until a Telnet connection is established. Telnet will also inform you if a connection actually breaks, so it's useful for testing login parameters; hence, if you have computer network or internet connection issues, you can try to log in via Telnet and test continuity.

Quite a lot of web netizens and webmasters possibly like yourself have never used a Telnet client (unless you're one of those geeky types). Network technicians will use it to debug routers, "troubleshoot" networks, or talk to their beloved pet web server. It can also be used for more … fun stuff.

Figure 12.11 is an example of a Telnet and SSH Client (in this case PuTTY[18]) running in SSH (which stands for Secure SHell) on Windows XP. The Telnet session is connecting to an online Telnet server, nicknamed Pallas, running a public SLASH'EM ("SuperLotsa Added Stuff Hack—Extended Magic") NetHack game server on a Linux box.[19] If you fancy playing a game of NetHack/SLASH'EM, you can easily type in commands and play a Dungeons-&-Dragons-like adventure, using your keyboard to move your player character within a graphical dungeon (but don't expect mind-blowing 3D graphics!).

[18] http://www.chiark.greenend.org.uk/~sgtatham/putty/
[19] http://slashem.crash-override.net/

www.it-ebooks.info

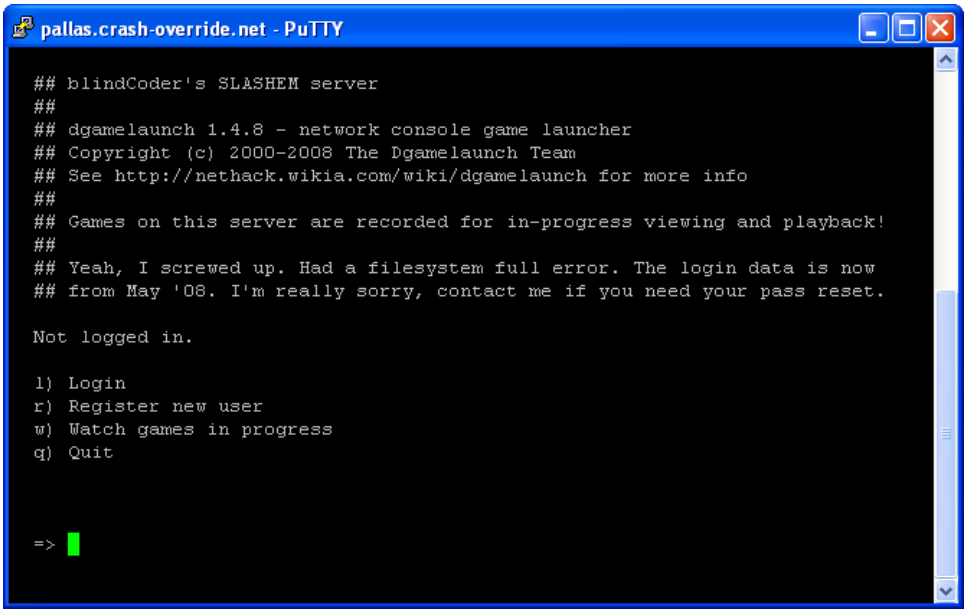
pallas.crash-override.net - PuTTY
blindCoder's SLASHEM server
##
dgamelaunch 1.4.8 - network console game launcher
Copyright (c) 2000-2008 The Dgamelaunch Team
See http://nethack.wikia.com/wiki/dgamelaunch for more info
##
Games on this server are recorded for in-progress viewing and playback!
##
Yeah, I screwed up. Had a filesystem full error. The login data is now
from May '08. I'm really sorry, contact me if you need your pass reset.
Not logged in.
l) Login
r) Register new user
w) Watch games in progress
q) Quit
=>

Figure 12.11. What an established Telnet session looks like

Usually, the Telnet protocol uses port 23, and Windows also offers a basic Telnet client called HyperTerminal, but it doesn't tend to be as comprehensive or simple to use as PuTTY. Telnet is an extremely powerful protocol, and is probably used by your web host if it's using Linux servers. Telnet only sends usernames and passwords in plain text. As a result, SSH (port 22) is generally used nowadays via a Telnet-like client for security reasons, as the data it sends can be encrypted with a special key—unlike standard the Telnet protocol.

Simple Mail Transfer Protocol (SMTP)

SMTP uses TCP port 25, allowing users to send email over the Internet. SMTP's primary role is to make sure email is forwarded to a POP3 server; however, it can only transmit plain text, not binary attachments like pictures, video, or sound clips.

Post Office Protocol (POP)—Version 3

POP3 is a mail protocol that uses TCP port 110, and stores email until delivery is required. POP3 transfers mail files from the mail server to an email client/program. Typical use would be a home user retrieving mail from a webmail service such as Gmail, or email clients like Mozilla Thunderbird and Mutt.

www.it-ebooks.info

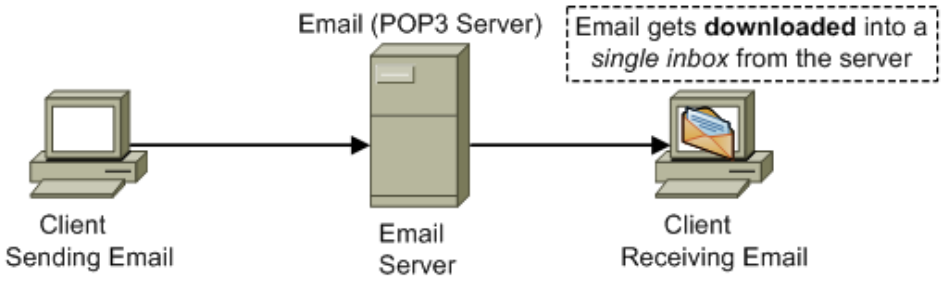
Email (POP3 Server)
Email gets **downloaded** into a *single inbox* from the server
Client Sending Email
Email Server
Client Receiving Email

Figure 12.12. Email client downloading mail from a POP3 server

Internet Messaging Access Protocol (IMAP)—Version 4

IMAP allows a client to access mail on a server, and performs various functions relating to mail within mailboxes residing on servers. Mail is sent via SMTP and retrieved using either POP3 or IMAP—the more complex protocol. In addition, IMAP is compatible with Multipurpose Internet Mail Extension (MIME). MIME allows transmission of multimedia files and binary data via a TCP/IP network.

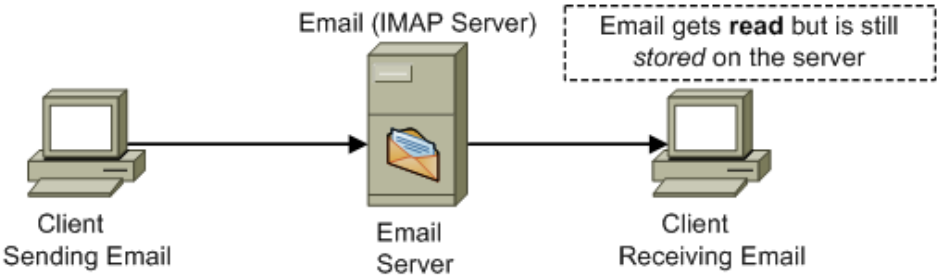
Email (IMAP Server)
Email gets **read** but is still *stored* on the server
Client Sending Email
Email Server
Client Receiving Email

Figure 12.13. Email client accessing mail from an IMAP server

An advantage of IMAP over POP3 is that it doesn't automatically download emails from the email server. Instead, you can connect to the server from various locations, and look at your mail messages before choosing to download them.

Synchronous/Asynchronous and Duplex/Simplex Data Transmission

Data travels using different methods.

www.it-ebooks.info

Synchronous and Asynchronous Transmission

Synchronous data transmission sends signals in a precise time relation using a clocking system, thus the transmission is synchronized.

Asynchronous data transmission is when the end of the transmission of one character initiates the transmission of the next.

Duplex and Simplex Transmission

Duplex transmission can transmit data in both directions at the same time, whereas half-duplex transmission can only do so one way at a time (but in both directions). An example of full-duplex is a telephone conversation where both participants can talk at the same time during conversation. In contrast, a simple CB radio transmits in half-duplex.

Simplex transmission, on the other hand, can only transmit in one direction; for example, a TV or radio broadcast.

Summary

You will have now learned that the seven layers of the OSI Model are efficient in defining how and where a network protocol will conduct its business. In addition, you'll know that the four-layered TCP/IP is the de facto protocol stack (suite of protocols) used for the Internet.

Furthermore, you're now aware that you've been using many of these protocols already without a second thought as to how they function in the background. I hope you now feel more confident in the event a web techie ever asks you a question regarding TCP/IP, FTP, Telnet, DNS, SNMP, HTTP, POP3, or the like—you should now be well-armed and ready.

"May the power of the Network be with you!"

www.it-ebooks.info